Scheduling in the Random-Order Model

Susanne Albers[1] and Maximilian Janke[2]

[1]Department of Computer Science, Technical University of Munich, albers@in.tum.de
[2]Department of Computer Science, Technical University of Munich, janke@in.tum.de

Abstract

This paper studies Makespan Minimization in the random-order model. Formally, jobs, specified by their processing times, are presented in a uniformly random order. An online algorithm has to assign each job permanently and irrevocably to one of m parallel and identical machines such that the expected time it takes to process them all, the makespan, is minimized.

We give two deterministic algorithms. First, a straightforward adaptation of the semi-online strategy LightLoad [3] provides a very simple algorithm retaining its competitive ratio of 1.75. A new and sophisticated algorithm is 1.535-competitive. These competitive ratios are not only obtained in expectation but, in fact, for all but a very tiny fraction of job orders.

Classically, online makespan minimization only considers the worst-case order. Here, no competitive ratio below 1.885 for deterministic algorithms and 1.581 using randomization is possible. The best randomized algorithm so far is 1.916-competitive. Our results show that classical worst-case orders are quite rare and pessimistic for many applications. They also demonstrate the power of randomization when compared much stronger deterministic reordering models.

We complement our results by providing first lower bounds. A competitive ratio obtained on nearly all possible job orders must be at least 1.257. This implies a lower bound of 1.043 for both deterministic and randomized algorithms in the general model.

1 Introduction

We study one of the most basic scheduling problems, the classic problem of makespan minimization. For the classic makespan minimization problem one is given an input set $\mathcal{J}$ of n jobs, which have to be scheduled onto m identical and parallel machines. Preemption is not allowed. Each job $J \in \mathcal{J}$ runs on precisely one machine. The goal is to find a schedule minimizing the *makespan*, i.e. the last completion time of a job. This problem admits a long line of research and countless practical applications in both, its offline variant see e.g. [29, 31] and references therein, as well as in the online setting studied in this paper.

In the online setting jobs are revealed one by one and each has to be scheduled by an online algorithm A immediately and irrevocably without knowing the sizes of future jobs. The makespan of online algorithm A, denoted by $A(\mathcal{J}^{\sigma})$, may depend on both the job set $\mathcal{J}$ and the job order σ. The optimum makespan $\text{OPT}(\mathcal{J})$ only depends on the former. Traditionally, one measures the performance of A in terms of competitive analysis. The input set $\mathcal{J}$ as well as the job order σ are chosen by an adversary whose goal is to maximize the ratio $\frac{A(\mathcal{J}^{\sigma})}{\text{OPT}(\mathcal{J})}$. The maximum ratio,

1

$c = \sup_{\mathcal{J},\sigma} \frac{A(\mathcal{J}^\sigma)}{\text{OPT}(\mathcal{J})}$, is the *(adversarial) competitive ratio.* The goal is to find online algorithms obtaining small competitive ratios.

In the classical secretary problem the goal is to hire the best secretary out of a linearly ordered set S of candidates. Its size n is known. Secretaries appear one by one in a uniformly random order. An online algorithm can only compare secretaries it has seen so far. It has to decide irrevocably for each new arrival whether this is the single one it wants to hire. Once a candidate is hired, future ones are automatically rejected even if they are better. The algorithm fails unless it picks the best secretary. Similar to makespan minimization this problem has been long studied, see [19, 22, 23, 32, 41, 43, 44] and references therein.

This paper studies a makespan minimization under the input model of the secretary problem. The adversary determines a job set of known size n. Similar to the secretary problem, these jobs are presented to an online algorithm A one by one in a uniformly random order. Again, A has to schedule each job without knowledge of the future. The expected makespan is considered. The *competitive ratio in the secretary (or random-order) model* $c = \sup_{\mathcal{J}} \mathbf{E}_\sigma \left[\frac{A(\mathcal{J}^\sigma)}{\text{OPT}(\mathcal{J})} \right]$ is the maximum ratio between the expected makespan of A and the optimum makespan. The goal is again to obtain small competitive ratios.

We propose the term *secretary model*, first used in [48], to set this model apart from the model studied by the same authors in [5] where n, the number of jobs, is not known in advance. Not knowing n is quite restrictive and has never been considered in any other work on scheduling with random-order arrival [26, 48, 49]. For the adversarial model such information is useless.

Similar frameworks received a lot of recent attention in the research community sparking the area of random-order analysis. Random-order analysis has been successfully applied to numerous problems such as matching [27, 33, 35, 45], various generalizations of the secretary problem [7, 22, 23, 32, 41, 43], knapsack problems [8], bin packing [39], facility location [46], packing LPs [40], convex optimization [30], welfare maximization [42], budgeted allocation [47] and recently scheduling [5, 26, 48, 49].

For makespan minimization the role of randomization is poorly understood. The lower bound of 1.581 from [11, 51] is considered pessimistic and exhibits quite a big gap towards the best randomized ratio of 1.916 from [2].

The upper bound of 1.535 in this paper demonstrates surprising power when it comes to randomization in the input order. The power of reordering has been studied by Englert et. al. [20]. Their lower bound considers online algorithms, which are able to look-ahead and rearrange almost all of the input sequence in advance. Their only disadvantage is that such rearrangement is deterministic. Englert et. al. show that these algorithms can not be better than 1.466-competitive for general m. This is quite close to our upper bound of 1.535, given that the algorithm involved has neither look-ahead nor control over the arrangement of the sequence.

A main consequence of the paper is that random-order arrival allows to beat the lower bound of 1.581 for randomized adversarial algorithms. This formally sets this model apart from the classical adversarial setting even if randomization is involved.

Previous work: Online makespan minimization and variants of the secretary problem have been studied extensively. We only review results most relevant to this work beginning with the traditional adversarial setting. For m identical machines, Graham [29] showed 1966 that the greedy strategy, which schedules each job onto a least loaded machine, is $\left(2 - \frac{1}{m}\right)$-competitive. This was subsequently improved in a long line of research [1, 9, 25, 34] leading to the currently best competitive ratio by Fleischer and Wahl [24], which approaches 1.9201 for $m \to \infty$. Chen et al. [12] presented

2

an algorithm whose competitive ratio is at most $(1 + \varepsilon)$-times the optimum one, though the actual ratio remains to be determined. For general m, lower bounds are provided in [10, 21, 28, 50]. The currently best bound is due to Rudin III [50] who shows that no deterministic online algorithm can be better than 1.88-competitive.

The role of randomization in this model is not well understood. The currently best randomized ratio of 1.916 [2] barely beats deterministic guarantees. In contrast, the best lower bound approaches $\frac{e}{e-1} > 1.581$ for $m \to \infty$ [11, 51]. There has been considerable research interest in tightening the gap.

Recent results for makespan minimization consider variants where the online algorithm obtains extra resources. There is the semi-online setting where additional information on the job sequence is given in advance, like the optimum makespan [36] or the total processing time of jobs [3, 13, 37, 38]. In the former model the optimum competitive ratio lies in the interval $[1.333, 1.5]$, see [36], while for the latter the optimum competitive ratio is known to be 1.585 [3, 38]. Taking this further, the advice complexity setting allows the algorithm to receive a certain number of advice bits from an offline oracle [4, 18, 37].

The work of Englert et. al. [20] is particularly relevant as they, too, study the power of reordering. Their algorithm has a buffer, which can reorder the sequence 'on the fly'. They prove that a buffer size linear in m suffices to be 1.466-competitive. Their lower bound shows that this result cannot be improved for any sensible buffer size.[1]

The secretary problem is even older than scheduling [23]. Since the literature is vast, we only summarize the work most relevant to this paper. Lindley [44] and Dynkin [19] first show that the optimum strategy finds the best secretary with probability $1/e$ for $n \to \infty$. Recent research focusses on many variants, among others generalizations to several secretaries [6, 41] or even matroids [7, 22, 43]. A modern version considers adversarial orders but allows prior sampling [14, 32]. Related models are prophet inequalities and the game of googol [15, 16].

So far, little is known for scheduling in the secretary model. Osborn and Torng [49] prove that Graham's greedy strategy is still not better than 2-competitive for $m \to \infty$. In [5] we studied the very restricted variant where n, the number of jobs, is not known in advance and provide a 1.8478-competitive algorithm and first lower bounds. Here, most common techniques, e.g. sampling, do not work. We are the only ones who ever considered this restriction. Molinary [48] studies a very general scheduling problem. His algorithm has expected makespan $(1 + \varepsilon)\text{OPT} + O(\log(m)/\varepsilon)$, but its random-order competitive ratio is not further analyzed. Göbel et al. [26] study a scheduling problem on a single machine where the goal is to minimize weighted completion times. Their competitive ratio is $O(\log(n))$ whereas they show that the adversarial model allows no sublinear competitive ratio.

Our contribution: We study makespan minimization for the secretary (or random-order) model in depth. We show that basic sampling ideas allow to adapt a fairly simple algorithm from the literature [3] to be 1.75-competitive. A more sophisticated algorithm vastly improves this competitive ratio to 1.535. This beats all lower bounds for adversarial scheduling, including the bound of 1.582 for randomized algorithms.

Our main results focus on a large number of machines, $m \to \infty$. This is in line with most recent adversarial results [2, 24] and all random-order scheduling results [5, 26, 48, 49]. While adversarial

---

[1]A buffer size of n would not be sensible since it reverts to the offline problem, which admits a PTAS [31]. Their lower bound holds for any buffer size $b(n)$, depending on the input size n, if $n - b(n)$ is unbounded. Such a buffer can already hold almost all, say any fraction, of the input sequence.

guarantees are known to improve for small numbers of machines, nobody has ever, to the best of our knowledge, explored guarantees for random-order arrival on a small number of machines. We prove that our simple algorithm is $\left(1.75 + O\left(\frac{1}{\sqrt{m}}\right)\right)$-competitive. Explicit bounds on the hidden term are given as well as simulations, which indicate good performance in practice. This shows that the focus of contemporary analyses on the limit case is sensible and does not hide unreasonably large additional terms.

All results in this paper abide to the stronger measure of *nearly competitiveness* from [5]. An algorithm is required to achieve its competitive ratio not only in expectation but on nearly all input permutations. Thus, input sequences where it is not obtained can be considered extremely rare and pathological. Moreover, we require worst-case guarantees even for such pathological inputs. This seems quite relevant to practical applications, where we do not expect fully random inputs. Both algorithms in this paper hold up to this stronger measure of nearly competitiveness.

A basic approch in random-order models relies on sampling; a small part of the input is used to predict the rest. Sampling allows us to include techniques from semi-online and advice scheduling with two further challenges. On the one hand, the advice is imperfect and may be, albeit with low probability, totally wrong. On the other hand, the advice has to be learned, rather than being available right from the start. In the beginning 'mistakes' cannot be avoided. This makes it impossible to adapt better semi-online algorithms than LightLoad, namely [13, 37, 38] to our model. These algorithms need to know the total processing volume right from the start. Note that the advanced algorithm in this paper out-competes the optimum competitive ratio of 1.585 these semi-online algorithms can achieve [1, 38]. We conjecture that this is not possible for algorithms that solely use sampling.

Algorithms that can only use sampling are studied in a modern variant of the secretary problem [14, 32]. First, a random sample is observed, then the sequence is treated in adversarial order. The analysis of LightLoad carries over to such a model without changes. The 1.535-competitive algorithm does not maintain its competitive ratio in such a model.

The 1.535-competitive main algorithm is based on a modern point of view, which, analogous to kernelization, reduces complex inputs to sets of critical jobs. A set of critical jobs is estimated using sampling. Critical jobs impose a lower bound on the optimum makespan. If the bound is high, an enhanced version of Graham's greedy strategy suffices; called the Least-Loaded-Strategy. Else, it is important to schedule critical jobs correctly. The Critical-Job-Strategy, based on sampling, estimates the critical jobs and schedules them ahead of time. An easy heuristic suffices, due to uncertainty involved in the estimates. Uncertainty poses not only the main challenge in the design of the Critical-Job-Strategy. On a larger scale, it also makes it hard to decide, which of the two strategies to use. Sometimes the Critical-Job-Strategy is chosen wrongly. These cases comprise the crux of the analysis and require using random-order arrival in a novel way beyond sampling.

The analyses of both algorithms follows three steps. First, adversarial analyses give worst-case guarantees and take care of *simple job sets*, which lack structure to be exploited via random reordering. Intuitively, random sequences have certain properties, like being not 'ordered'. A second step formalizes this, introducing stable orders. Non-stable orders are rare and negligible. Reducing to stable orders yields a natural semi-online setting. Third, we analyze our algorithm in this semi-online setting. See Figure 3 for a lay of the land.

The paper concludes with lower bounds for the secretary model. No algorithm, deterministic or randomized, is better than nearly 1.257-competitive. This immediately yields a lower bound of 1.043 in the general secretary model, too.

4

2 Notation

Almost all notations relevant in scheduling depend on the input set $\mathcal{J}$ or on the ordered input sequence $\mathcal{J}^\sigma$. We use the notation $[\mathcal{J}]$ and $[\mathcal{J}^\sigma]$ to indicate such dependency, for example $L[\mathcal{J}]$ and $L_\varphi[\mathcal{J}^\sigma]$. If such dependency needs not be mentioned, for example if the sequence $\mathcal{J}^\sigma$ is fixed, we drop this appendage, simply writing L and L_φ. Similarly, we write OPT for $\mathrm{OPT}(\mathcal{J})$. If we focus on the job order σ whilst the dependency of the job set $\mathcal{J}$ does not deserve mention, the notation $[\sigma]$ instead of $[\mathcal{J}^\sigma]$ is used. We could for example write $L_\varphi[\sigma]$.

3 A strong measure of random-order competitiveness

Consider a set of n jobs $\mathcal{J} = \{J_1, \ldots, J_n\}$ with non-negative sizes $p_1, \ldots, p_n$ and let S_n be the group of permutations of the integers from 1 to n. We consider S_n a probability space under the uniform distribution, i.e. we pick each permutation with probability $1/n!$. Each permutation $\sigma \in S_n$, called an *order*, gives a *job sequence* $\mathcal{J}^\sigma = J_{\sigma(1)}, \ldots, J_{\sigma(n)}$. Recall that traditionally an online algorithm A is called *c-competitive* for some $c \geq 1$ if we have for all job sets $\mathcal{J}$ and job orders σ that $A(\mathcal{J}^\sigma) \leq c\mathrm{OPT}(\mathcal{J})$. We call this the *adversarial model*.

In the secretary model we consider the expected makespan of A under a uniformly chosen job order, i.e. $A^{\mathrm{rom}} = \mathbf{E}_{\sigma \sim S_n}[A(\mathcal{J}^\sigma)] = \frac{1}{n!} \sum_{\sigma \in S_n} A(\mathcal{J}^\sigma)$, rather than the makespan achieved in a worst-case order. The algorithm A is *c-competitive in the secretary model* if $A^{\mathrm{rom}}(\mathcal{J}) \leq c\mathrm{OPT}(\mathcal{J})$ for all input sets $\mathcal{J}$.

This model tries to lower the impact of particularly badly ordered sequences by looking at competitive ratios only in expectation. Interestingly, the scheduling problem allows for a stronger measure of random-order competitiveness for large m, called *nearly competitiveness* [5]. One requires the given competitive ratio to be obtained on nearly all sequences, not only in expectation, as well as a bound on the adversarial competitive ratio as well. We recall the definition and the main fact, that an algorithm is already c-competitive in the secretary model if it is nearly c-competitive.

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 $\varepsilon > 0$, we can find $m(\varepsilon)$ such that for all machine numbers $m \geq m(\varepsilon)$ and all job sequences $\mathcal{J}$ there holds $\mathbf{P}_{\sigma \sim S_n}[A(\mathcal{J}^\sigma) \geq (c + \varepsilon)\mathrm{OPT}(\mathcal{J})] \leq \varepsilon$.*

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 adversarial competitive ratio of A. Given $\delta > 0$ we need to show that we can choose m large enough such that our algorithm is $(c+\delta)$-competitive in the random-order model. For $\varepsilon = \frac{\delta}{C-c+1}$ choose m large enough such that $P_\varepsilon(\mathcal{J}) = \mathbf{P}_{\sigma \sim S_n}[C_A(\mathcal{J}^\sigma) \geq (c + \varepsilon)\mathrm{OPT}(\mathcal{J})] \leq \varepsilon$ holds for every input sequence $\mathcal{J}$. Then we have for every input sequence $\mathcal{J}$ that

$$C^{\mathrm{rom}}(\mathcal{J}) \leq (1 - P_\varepsilon(\mathcal{J})) \cdot (c + \varepsilon)\mathrm{OPT}(\mathcal{J}) + P_\varepsilon(\mathcal{J}) \cdot C \cdot \mathrm{OPT}(\mathcal{J}) \leq ((1 - \varepsilon)(c + \varepsilon) + \varepsilon C)\mathrm{OPT}(\mathcal{J})$$
$$\leq (c + \delta(C - c + 1))\mathrm{OPT}(\mathcal{J}) = (c + \delta)\mathrm{OPT}(\mathcal{J}). \qquad \square$$

4 Basic properties

Given an input sequence $\mathcal{J}^\sigma = J_{\sigma(1)}, \ldots J_{\sigma(n)}$ and $0 < \varphi \le 1$, we consider the *load estimate* $L_\varphi = L_\varphi[\mathcal{J}^\sigma] = \frac{1}{\varphi m} \sum_{\sigma(t) \le \varphi n} p_t$, which is φ^{-1}-times the average load (in any schedule) after the first φn jobs have been assigned. We are particularly interested in the *average load* $L = L[\mathcal{J}] = L_1[\mathcal{J}^\sigma]$, which is a lower bound for OPT. The value L_φ for smaller φ is a guess for L, which can be made by an online algorithm after a φ-fraction of the input-sequence has been observed. Given $t > 0$, let $p_{\max}^t = \max(p_{t'} \mid t' < t + 1)$ be the size of the largest among the first $\lceil t \rceil$ jobs. In particular, $p_{\max} = p_{\max}^n$, the *size of the largest jobs*, is again an important lower bound for OPT.

Proposition 3. *We have the following lower bounds for the optimum makespan:*

- $p_{\max} \le \text{OPT}$
- $L \le \text{OPT}$

Proof. The first bound follows from observing that any schedule must, in particular, schedule the largest job on some machine whose load thus is at least $p_{\max}$. For the second bound one observes that the makespan, the maximum load of a machine in any given schedule, cannot be less than L, the average load of all machines. $\qquad\square$

Let us consider any fixed (ordered) job sequence $\mathcal{J}^\sigma = J_{\sigma(1)}, \ldots J_{\sigma(n)}$ and any (deterministic) algorithm that assigns these jobs to machines. We begin with some fundamental observations.

Lemma 4. *Let $\varphi > 0$ and $t \le \varphi n$. Then the k-th least loaded machine at time t has load at most $\frac{m}{m-k+1} \varphi L_\varphi$. In particular, its load is at most $\frac{m}{m-k+1} L$.*

Proof. Let L^t be the sum of all loads at time t. Since this is the same as the sum of all processing times of jobs arriving at time t, we have $L^t \le \sum_{\sigma(t') \le t} p_{t'} \le \varphi m \frac{1}{\varphi m} \sum_{\sigma(t') \le \varphi n} p_{t'} = \varphi m L_\varphi$. Let l be the load of the k-th least loaded machine at time t. Per definition $m - k + 1$ machines had at least that load. Thus $(m - k + 1)l \le L^t \le \varphi m L_\varphi$ or, equivalently, $l \le \frac{m}{m-k+1} \varphi L_\varphi$. $\qquad\square$

Consider the value $R(\mathcal{J}) = \min\{\frac{L}{p_{\max}}, 1\}$, which measures the complexity of the input set independent of its order. Informally, a smaller value $R(\mathcal{J})$ makes the job set easier to be scheduled but less suited to reordering arguments. Later, sets with a small value $R(\mathcal{J})$ need to be treated separately. The following proposition is both interesting for its implication on general sequences and, particularly, simple sequences with $R(\mathcal{J})$ small.

Proposition 5. *If any job J is scheduled on the k-th least loaded machine, the load of said machine does not exceed $\left(\left(\frac{m}{m-i+1} \right) R(\mathcal{J}) + 1 \right) OPT(\mathcal{J})$ afterwards.*
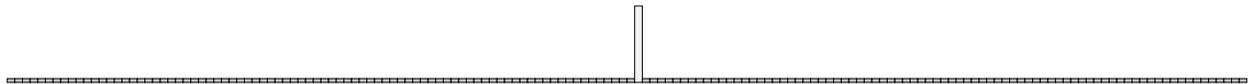

Figure 1: A surprisingly difficult sequence for random-order arguments. The big job carries most of the processing volume. Other jobs are negligible. Thus, all permutations look basically the same. Note that for such a 'simple' sequence $R(\mathcal{J})$ is small.

Proof. Let l be the load of the i-th least loaded machine before J is scheduled. Then $l \leq \frac{m}{m-k+1}L$ by Lemma 4. Since J had size at most $p_{\max}$, the load of the machine it was scheduled on won't exceed

$$l + p_{\max} \leq \frac{m}{m-k+1}L + p_{\max} \leq \frac{m}{m-k+1}\frac{L}{\max(L,p_{\max})}\text{OPT} + \text{OPT} = \left(\left(\frac{m}{m-k+1}\right)R(\mathcal{J}) + 1\right)OPT. \qquad \square$$

Thus, if an algorithm avoids a constant fraction of most loaded machines, its competitive ratio is bounded and approaches 1 as $R(\mathcal{J}) \to 0$.

We call a vector $(\tilde{l}_M^t)$ indexed over all machines M and all times $t = 0, \dots, n$ a *pseudo-load* if $\tilde{l}_M^t \geq l_M^t$ for any time t and machine M. We introduce such a pseudo-load in the analysis of our main algorithm. Let $\tilde{L} = \sup_t \frac{1}{m}\sum_M \tilde{l}_M^t$ be the *maximum average pseudo-load* and, again, consider $\tilde{R}(\mathcal{J}) = \min\{\frac{\tilde{L}}{p_{\max}}, \frac{\tilde{L}}{L}\} = R(\mathcal{J})\frac{\tilde{L}}{L}$. The following observation is immediate.

Lemma 6. *We have $\tilde{L} \geq L$ and $R(\mathcal{J}) \leq \tilde{R}(\mathcal{J})$.*

Proof. Indeed, $L = \frac{1}{m}\sum_M l_M^n \leq \frac{1}{m}\sum_M \tilde{l}_M^n \leq \tilde{L}$. This already implies $R(\mathcal{J}) \leq \tilde{R}(\mathcal{J})$. $\qquad \square$

It will be important to note that Lemma 4 and Proposition 5 generalize to pseudo-loads. Since the proofs stay almost the same, we do not include them in the main body of the paper but leave them to Appendix A for completeness.

Lemma 7. *Let $\varphi > 0$ and $t \leq \varphi n$. Then the machine with the k-th least pseudo-load at time t had pseudo-load at most $\frac{m}{m-k+1}\tilde{L}$.*

Proposition 8. *If job $J_{\sigma(t+1)}$ is scheduled on the machine M with i-th smallest pseudo-load $\tilde{l}_M^t$ at time t, then, afterwards, its load l_M^{t+1} does not exceed $\left(1 + \left(\frac{m}{m+1-i}\right)\tilde{R}(\mathcal{J})\right)OPT(\mathcal{J})$.*

4.1 Sampling and the Load Lemma

Our model is particularly suited to sampling. Given a job set $\mathcal{J}$, we call a subset $\mathcal{C} \subseteq \mathcal{J}$ a *job class*. Consider any job order $\sigma \in S_n$. For $0 < \varphi \leq 1$, let $n_{\mathcal{C},\varphi}[\sigma]$ denote the number of jobs in $\mathcal{C}$ arriving till time φn, i.e. $n_{\mathcal{C},\varphi}[\sigma] = |\{J_{\sigma(i)} \mid J_{\sigma(i)} \in \mathcal{C} \wedge \sigma(i) \leq \varphi n\}|$. Let $n_{\mathcal{C}} = n_{\mathcal{C},1}[\sigma] = |\mathcal{C}|$ be the total number of jobs in $\mathcal{C}$. The following is a consequence of Chebyshev's inequality. The proof is left to Appendix A.

Proposition 9. *Let $\mathcal{C} \subset \mathcal{J}$ be a job class for a job set $\mathcal{J}$ of cardinality at least m. Given $\varphi > 0$ and $E \geq 0$ we have*

$$\mathbf{P}_{\sigma \sim S_n}\left[\left|\varphi^{-1}n_{\mathcal{C},\varphi}[\sigma] - n_{\mathcal{C}}\right| \geq E\right] \leq \frac{n_{\mathcal{C}}}{\varphi(E - 1/m)^2}.$$

A basic lemma in random-order scheduling is the Load Lemma from [5], which allows a good estimate of the average load under very mild assumptions on the job set. Here, we introduce a more general version. It is all we need to adapt the semi-online algorithm LightLoad from the literature to the secretary model.

Lemma 10. *[Load Lemma [5]] Let $R_{\mathrm{low}} = R_{\mathrm{low}}(m) > 0$, $1 \geq \varphi = \varphi(m) > 0$ and $\varepsilon = \varepsilon(m) > 0$ be three functions such that $\varepsilon^{-4}\varphi^{-1}R_{\mathrm{low}}^{-1} = o(m)$. Then there exists a variable $m(R_{\mathrm{low}}, \varphi, \varepsilon)$ such that we have for all $m \geq m(R_{\mathrm{low}}, \varphi, \varepsilon)$ and all job sets $\mathcal{J}$ with $R(\mathcal{J}) \geq R_{\mathrm{low}}$ and $|\mathcal{J}| \geq m$:*

$$\mathbf{P}_{\sigma \sim S_n}\left[\left|\frac{L_\varphi[\mathcal{J}^\sigma]}{L[\mathcal{J}]} - 1\right| \geq \varepsilon\right] < \varepsilon.$$
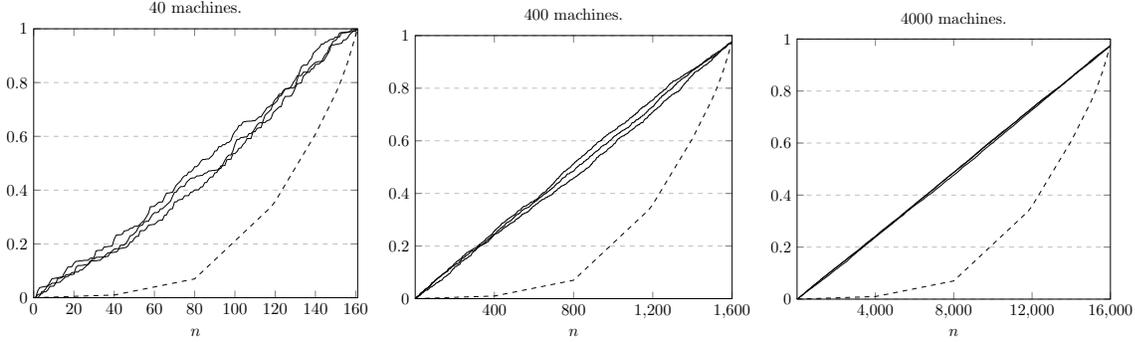
Figure 2: A graphic depicting the average load over time on the classical lower bound sequence from [1] for 40, 400 and 4000 machines. The dashed line corresponds to the original adversarial order. The three solid lines corresponding to random permutations clearly approximate a straight line. Thus, sampling allows to predict the (final) average load.

We sketch the proof, leaving the details to Appendix A since it is technical and a slight generalization of the one found in [5]. We use geometric rounding so that we only have to deal with countably many possible job sizes. Now, jobs of any given size p form a job class $\mathcal{C} = \mathcal{C}_p$. Using Proposition 9, we can relate their actual cardinality $n_{\mathcal{C}}$ with the φ-estimate $n_{\mathcal{C},\varphi}$. Putting everything together yields the Load Lemma, which compares the load L and the load estimate L_φ. The lemma relies intrinsically on the lower bound R_{low} for $R(\mathcal{J})$. Consider a job set $\mathcal{J}$ like the one in Figure 1, only one job carries all the load while there are lots of other jobs with size zero (or negligible size $\epsilon > 0$). Then $R(\mathcal{J}) = \frac{1}{m}$ and a statement as in Lemma 10 could not be true for $\varepsilon < \min(1, \varphi^{-1} - 1)$ since $L_\varphi \in \{0, \varphi^{-1}L\}$.

Figure 2 shows the behavior of the average load on three randomly chosen permutations of a classical input sequence. As predicted, this average load approaches a straight line for large number of machines. The Load Lemma is an important theoretical tool but only provides asymptotic guarantees. In Section 4.2 we explore practical guarantees for small numbers of machines.

4.2   A simple 1.75-competitive algorithm

We modify the semi-online algorithm LightLoad from the literature to obtain a very simple nearly 1.75-competitive algorithm. For any $0 \le t \le n$ let M_{mid}^t be a machine having the $\lfloor m/2 \rfloor$-lowest load at time t, i.e. right before job J_{t+1} is scheduled. Let l_{mid}^t be its load and let l_{low}^t be the smallest load of any machine. We recall the algorithm LightLoad[L_{guess}] from Albers and Hellwig [3], where the parameter L_{guess} is a guess for L.

Algorithm 1 The (semi-online) algorithm LightLoad[L_{guess}] [3].

1: *Let J_t be the job to be scheduled and let p_t be its size.*
2: **if** $l_{\text{low}}^{t-1} \le 0.25 L_{\text{guess}}$ **or** $l_{\text{mid}}^{t-1} + p_t > 1.75 L_{\text{guess}}$ **then**
3:     Schedule J_t on any least loaded machine;
4: **else** schedule J_t on M_{mid}^{t-1};

LightLoad[L] has been analyzed in the setting where the average load L is known in advance,

8

i.e. with fixed parameter $L_{\text{guess}} = L$. Albers and Hellwig obtain the following:

Theorem 11 ([3])**.** LightLoad[L] *is adversarially* 1.75*-competitive, i.e. for every job sequence* $\mathcal{J}^\sigma$ *with average load* $L = L[\mathcal{J}]$ *there holds* LightLoad$[L](\mathcal{J}^\sigma) \leq 1.75\text{OPT}(\mathcal{J})$.

The proof from [3] is complicated and not repeated in this paper. We need to deal with more general guesses L_{guess} that are slightly off. The following corollary is derived from Theorem 11 by enlarging the input sequence.

Corollary 12. *Let* $\mathcal{J}^\sigma$ *be any (ordered) input sequence and let* $L_{\text{guess}} \geq L[\mathcal{J}]$*. Then the makespan of* LightLoad$[L_{\text{guess}}]$ *is at most* $1.75 \cdot \max(L_{\text{guess}}, \text{OPT}(\mathcal{J}))$.

The idea of the proof is rather simple. We can add jobs to the end of the sequence $\mathcal{J}^\sigma$ such that for the resulting sequence $\mathcal{J}'^{\sigma'}$ there holds $\text{OPT}(\mathcal{J}') = \max(L_{\text{guess}}, \text{OPT}(\mathcal{J}))$. We then apply Theorem 11 to see that LightLoad$[L_{\text{guess}}]$ has cost at most $1.75 \cdot \max(L_{\text{guess}}, \text{OPT}(\mathcal{J}))$ on this sequence. Passing over to the prefix $\mathcal{J}^\sigma$ of $\mathcal{J}'^{\sigma'}$ cannot increase this cost. A technical proof is left to Appendix B for completeness.

We also need to deal with guesses L_{guess} that are totally of. Since LightLoad$[L_{\text{guess}}]$ only considers the least or the $\lfloor m/2 \rfloor$-th least loaded machine we get by Proposition 5:

Corollary 13. *For any (ordered) sequence* $\mathcal{J}^\sigma$ *and any value* L_{guess} *the makespan of* LightLoad$[L_{\text{guess}}]$ *is at most* $(1 + 2R(\mathcal{J}))\text{OPT}(\mathcal{J})$*. In particular, it is at most* 3OPT$(\mathcal{J})$.

Adapting LightLoad to the random-order model

Let $\delta = \delta(m) = 1/\log(m)$ be the *margin of error our algorithm allows*. We will see that our algorithm is $(1.75 + O(\delta))$-competitive. In fact, any function with $\delta(m) \in \omega(m^{-1/4})$ and $\delta(m) \in o_m(1)$ would do. Given an input sequence $\mathcal{J}^\sigma$ let $\hat{L}_{\text{pre}} = \hat{L}_{\text{pre}}[\mathcal{J}^\sigma] = \frac{L_{1/4}[\mathcal{J}^\sigma]}{1-\delta}$ be our *guess for* L. We use the index 'pre' since our main algorithm later will use a slightly different guess $\hat{L}$. In this section we consider the algorithm LightLoadROM = LightLoad$[\hat{L}_{\text{pre}}]$. Let us observe first that this is indeed an online algorithm, not only a semi-online algorithm as one might expect since the **if**-clause uses the guess $\hat{L}_{\text{pre}}$ before it is known.

Lemma 14. *The algorithm* LightLoadROM *can be implemented as an online algorithm.*

Proof. It suffices to note that the **if**-clause always evaluates to TRUE for $t < n/4$, i.e. before $L_{\text{guess}} = \hat{L}_{\text{pre}}$ is known. Indeed, in this case $l_{\text{low}} \leq 0.25L_{1/4} < 0.25\hat{L}_{\text{pre}}$ by Lemma 4. $\qquad\square$

We now prove the main theorem. Corollary 16 follows immediately from Lemma 2.

Theorem 15. *The algorithm* LightLoadROM *is nearly* 1.75*-competitive.*

Corollary 16. LightLoadROM *is* 1.75*-competitive in the secretary model for* $m \to \infty$.

9

Figure 3: The lay of the land of the analysis. The algorithm is $(c + O(\delta))$-competitive on simple and proper stable sequences. Only the small unstable remainder (hashed) is problematic. Dashed lines mark orbits under the action of the permutation group S_n. Simple sequences stay simple under permutation. Non-simple orbits have at most an δ-fraction, which is unstable (hashed). Thus, the algorithm is $(c + O(\delta))$-competitive with probability at least $1 - \delta$ after random permutation.

Proof of Theorem 15. Our analysis forms a triad, which outlines how we are going to analyze our more sophisticated 1.535-competitive algorithm later on as well.

Analysis basics: By Corollary 13 the algorithm LightLoadROM is adversarially 3-competitive. We call the input set $\mathcal{J}$ *simple* if $|\mathcal{J}| \leq m$ or $R[\mathcal{J}] < \frac{3}{8}$. If $|\mathcal{J}| \leq m$ every job is scheduled onto an empty machine, which is optimal. If $R[\mathcal{J}] < \frac{3}{8}$, Corollary 13 bounds the competitive ratio by $1 + 2R[\mathcal{J}] < 1.75$. We thus are left to consider non-simple, so called *proper*, job sets.

Stable job sequences: We call a sequence $\mathcal{J}^\sigma$ stable if $L \leq \hat{L}_{\mathrm{pre}} \leq \frac{1+\delta}{1-\delta}L$ holds true. By the Load Lemma, Lemma 10, the probability of the sequence $\mathcal{J}^\sigma$ being stable is at least $1 - \delta$ if we choose m large enough and $\mathcal{J}$ proper. Here we use that $\delta(m) = 1/\log(m) \in \omega(m^{-1/4})$.

Adversarial Analysis: By Corollary 12, the makespan of LightLoadROM on stable sequences is at most $1.75 \max(\hat{L}_{\mathrm{pre}}(\mathcal{J}), \mathrm{OPT}(\mathcal{J})) \leq 1.75\frac{1+\delta}{1-\delta}\mathrm{OPT}(\mathcal{J}) = \left(1.75 + \frac{3.5 \cdot \delta}{1-\delta}\right)\mathrm{OPT}(\mathcal{J})$.

Conclusion: Let $\varepsilon > 0$. Since $\delta(m) \to 0$, we can choose m large enough such that $\frac{3.5\delta(m)}{1-\delta(m)} \leq \varepsilon$. In particular $\mathbf{P}_{\sigma \sim S_n}[\mathrm{LightLoadROM}(\mathcal{J}^\sigma) \geq (c + \varepsilon)OPT(\mathcal{J})] \leq \delta(m) \leq \varepsilon$ since the only sequences where the inequality does not hold are proper but not stable. This concludes the second condition of nearly competitivity. $\qquad\square$

Why underestimating L is actually not as bad as one may think.

So far we were careful to choose our guess $\hat{L}_{\mathrm{pre}}$ in such a way that it is unlikely to underestimate L since this allowed us to prove results in a self-contained fashion, using Theorem 11 from [3] only as a black box. One should note that their analysis also allows us to tackle guesses $L_{\mathrm{guess}} < L$.

Lemma 17. *Let $\mathcal{J}^\sigma$ be any (ordered) input sequence and let $L_{\mathrm{guess}} = (1 - \delta)L[\mathcal{J}]$ for some $\delta \geq 0$. Then the makespan of $\mathrm{LightLoad}[L_{\mathrm{guess}}]$ is at most $1.75(1 + \delta)\mathrm{OPT}(\mathcal{J})$.*

Showing this lemma requires carefully rereading the analysis of Albers and Helwig [3]. We next describe how their analysis has to be adapted to derive Lemma 17.

How to adapt the proof from [3]. Consider any input sequence $J_1, \ldots J_n$. Using induction we may assume the result of the lemma to hold on the prefix $J_1, \ldots, J_{n-1}$. In [3] they argue that the algorithm remains 1.75-competitive if the least loaded machine had load at most $0.75L$ upon arrival of J_n. By a similar reasoning the less strict statement of Lemma 17 holds if the least loaded machine had load at most $0.75L + 1.75\delta L$ at that time. Thus we are left to consider the case that its load is $0.75L + 1.75\delta L + \epsilon L$ for some $0 < \epsilon < 0.25 - 1.75\delta$. Following the arguments [3], it suffices to show

that every machine received a job of size $0.5L + \epsilon L$. The statement of Lemma 1 in [3] needs to be weakened to 'At time t_{j_0} the least loaded machine had load at most $(0.25 + 1.75\delta)L$.' The proof of the lemma remains mostly the same. The only change occurs in the induction step. Here, the size of a job causing a machine to reach load $0.75L + \delta L + \epsilon L$ and, in addition, the corresponding decrease in potential is only $(1 - 1.75\delta)L$. Similarly, the statement of Lemma 2 needs to be refined to 'the j_0-th least loaded machine had load at most $(1.25 + 1.75\delta)L - \epsilon L = 1.75L_{\text{guess}} - 0.5L - \epsilon L$.' The proof of Lemma 2 stays the same. Using these modifications, the rest of the analysis of [3] can be applied to conclude the proof. $\qquad\square$

Theorem 18. *Let $\mathcal{J}^\sigma$ be any (ordered) input sequence. The makespan of* LightLoadROM *on $\mathcal{J}^\sigma$ is $1.75 \cdot \left(1 + \frac{|\hat{L}_{\text{pre}}[\mathcal{J}^\sigma] - L|}{L}\right)$* OPT.

Proof. On input permutation $\mathcal{J}^\sigma$ the competitive ratio of our algorithm is at most $1.75 + \frac{|L_{1/4} - L|}{L}$ by Corollary 12 if $L_{1/4} \geq L$. Else, recall that OPT $\geq L$ and apply Lemma 17. $\qquad\square$

Let us assume for simplicity that the input length n is divisible by 4. We can always add up to three jobs of size 0 to obtain such a result. This 'adding' can be simulated by an online algorithm. Recall that the *absolute mean deviation* of a random variable X that has nonzero expectation is defined as $\text{MD}[X] = \mathbf{E}\left[|X - \mathbf{E}[X]|\right]$ and its *normalized absolute mean deviation* is $\text{NMD}[X] = \frac{\mathbf{E}[|X - \mathbf{E}[X]|]}{\mathbf{E}[X]}$. In particular, $\text{NMD}[L_{1/4}] = \frac{\mathbf{E}_{\sigma \sim S_N}[|L_{1/4} - L|]}{L}$. From the previous theorem we obtain:

Theorem 19. *On input set $\mathcal{J}$ the competitive ratio of* LightLoadROM *in the random-order model is at most $1.75(1 + \text{NMD}(\hat{L}_{\text{pre}}))$. If $n \leq m$ or $R(\mathcal{J}) \leq \frac{3}{8}$, then* LightLoad$[L_{1/4}]$ *is already 1.75-competitive.*

Proof. The first statement follows from Theorem 18 by taking expected values. If $n \leq m$, the algorithm LightLoad$[L_{1/4}]$ places every job on a separate machine and is thus optimal. If $R(\mathcal{J}) \leq \frac{3}{8}$, it is 1.75-competitive by Corollary 13. $\qquad\square$

We will now provide estimates on $\text{NMD}(\hat{L}_{\text{pre}})$. Figure 4 depicts practical estimates, while our analysis will focus on theoretical bounds.

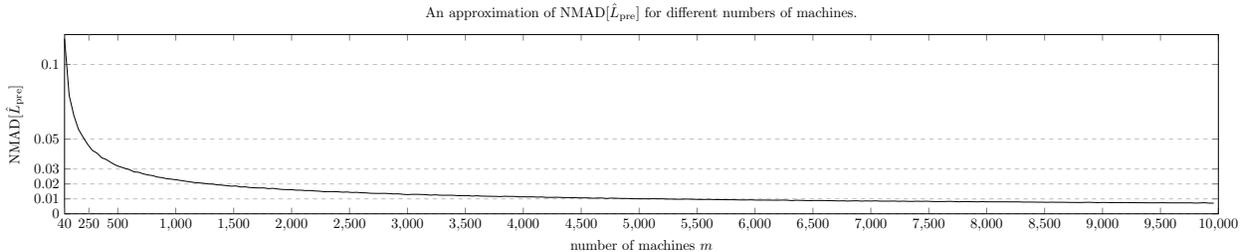An approximation of NMAD[$\hat{L}_{\text{pre}}$] for different numbers of machines.

Figure 4: The extra cost for small numbers of machines. The graph shows an estimation of $\text{NMD}(\hat{L}_{\text{pre}})$ on the lower bound sequence from [1] based on $10,000$ random samples. The curve indicates good performance of LightLoadROM in practice by Theorem 19.

Given any job set $\mathcal{J}$ of size $n > m$ and $R(\mathcal{J}) \geq \frac{3}{8}$ we are left to estimate this normalized standard deviation of $L_{1/4}$. One observes that $\text{NMD}[L_{1/4}]$ does not change if we scale all jobs by

a common factor $\lambda > 0$. By choosing $\lambda = L^{-1}$ we may wlog. assume that $L = 1$. In particular, $\mathrm{NMD}[L_{1/4}] = \mathrm{MD}[L_{1/4}]$. Now $R(\mathcal{J}) \geq \frac{3}{8}$ implies that all jobs have size at most 8/3. The following lemma allows us to reduce ourselves to particularly easy job instances.

Lemma 20. *Consider two jobs $J_a, J_b \in \mathcal{J}$ of sizes $p_a \leq p_b$ and $0 \leq \varepsilon \leq p_a$. If we set the size of J_a to $p_a - \varepsilon$ and the size of J_b to $p_b + \varepsilon$, then $\mathrm{MD}(L_{1/4})$ does not decrease.*

Proof. Consider $\mathrm{MD}(L_{1/4})(p_1, \ldots, p_n) = \mathbf{E}[|L_{1/4} - L|](p_1, \ldots, p_n)$ as a function on the job sizes $p_1, \ldots, p_n$. This function is convex since it is a convex combination of the convex functions $|L_{1/4}[\sigma] - L|$ for all $\sigma \in S_n$. $\square$

We apply the previous lemma to any pairs of $J_a, J_b \in \mathcal{J}$ of sizes $0 < p_a \leq p_b < 8/3$ to set either $p_a = 0$ or $p_b = 8/3$. We then repeat this process till all jobs but one last one have either size 0 or 8/3. So far at most $\lfloor \frac{3m}{8} \rfloor$ jobs have size 8/3 since by assumption $L = 1$. Using again the fact that $\mathrm{MD}[L_{1/4}]$ is convex in the size of jobs, setting the size p of this remaining job to at least one of the values 0 or 8/3 cannot decrease $\mathrm{MD}[L_{1/4}]$. Let us do so. This breaks the assumption that $L = 1$, which is why we consider MD instead of NMD. Let $K = K(n)$ be the number of jobs of size 8/3. Then K is either $\lfloor \frac{3m}{8} \rfloor$ or $\lfloor \frac{3m}{8} \rfloor + 1$. Let $X \sim \mathrm{HyperGeom}(n, K, n/4)$, in other words X corresponds to drawing $n/4$ elements without replacement from a population of size n that contains precisely K successes. Then $L_{1/4} = \frac{8X}{3m}$ for our modified job set. Since all modifications never caused $\mathrm{MD}[X]$ to decrease we have shown so far:

Lemma 21. *Let $\mathcal{J}$ be a job set of size $n > m$ with $R(\mathcal{J}) \leq \frac{3}{8}$. Then we can choose K either $\lfloor \frac{3m}{8} \rfloor$ or $\lfloor \frac{3m}{8} \rfloor + 1$ such that for $X \sim \mathrm{HyperGeom}(n, K, n/4)$ there holds $\mathrm{NMD}[L_{1/4}] \leq \frac{8}{3m}\mathrm{MD}[X]$.*

It is possible to evaluate the standard mean deviation of $X \sim \mathrm{HyperGeom}(n, K, n/4)$ directly using the techniques in [17]. Since such an analysis is quite complex we present a simpler proof, which yields somewhat worse bounds.

Lemma 22. *Let $X \sim \mathrm{HyperGeom}(n, K, n/4)$ and $Y \sim \mathrm{Bin}(K, 1/4)$ then $\mathrm{MD}(X) \leq \mathrm{MD}(Y)$.*

Proof. Indeed, both random variables correspond to K draws from a population of size n that contains $n/4$ successes. For X these draws occur without replacement, while for Y these are draws with replacement. Let X_i respectively Y_i be the respective random variable, which only considers the first i draws for $0 \leq i \leq K$. We can show via induction that the random variable $|Y_i - i/4|$ dominates $|X_i - i/4|$ by a case distinction on the possible values of $Y_{i-1} - (i-1)/4$ and $X_{i-1} - (i-1)/4$. Thus, the random variable $|Y - \mathbf{E}[Y]|$ dominates $|X - \mathbf{E}[X]|$. This implies $\mathrm{MD}(Y) = \mathbf{E}[|Y - \mathbf{E}[Y]|] \geq \mathbf{E}[|X - \mathbf{E}[X]|] = \mathrm{MD}(X)$. $\square$

Given $Y \sim \mathrm{Bin}(K, p)$, we are interested in $p = 1/4$, let $\mathrm{bin}(k, K, p) = P[Y = k] = \binom{K}{k}p^k(1 - p)^{K-k}$. Then we can evaluate the median deviation of Y using de Moivre's theorem.

Theorem 23 (de Moivre). $\mathrm{MD}(Y) = 2\lfloor pK + 1 \rfloor(1 - p)\mathrm{bin}(\lfloor pK + 1 \rfloor, K, p)$ *for $Y \sim \mathrm{Bin}(K, p)$.*

A proof of the theorem can be found in [17]. Consider $Y \sim \mathrm{Bin}(K, 1/4)$ and set $k = \lfloor K/4 + 1 \rfloor$.

Using Stirling's Approximation we derive

$$\text{MD}(Y) = \lfloor 2k \rfloor \cdot 3/4 \cdot \text{bin}\,(k, K, 1/4)$$

$$= (1 + o_K(1))\frac{3}{8}K\frac{K!}{k!(K-k)!}(1/4)^k(3/4)^{K-k}$$

$$= (1 + o_K(1))\frac{3}{8}K\frac{\sqrt{2\pi K}\left(\frac{K}{e}\right)^K}{\sqrt{2\pi k}\left(\frac{k}{e}\right)^k\sqrt{2\pi(K-k)}\left(\frac{K-k}{e}\right)^{K-k}}\left(\frac{1}{4}\right)^k\left(\frac{3}{4}\right)^{K-k}$$

$$= (1 + o_K(1))\frac{3}{8}K\sqrt{\frac{K}{2\pi k(K-k)}}\left(\frac{K}{4k}\right)^k\left(\frac{3}{4}\frac{K}{K-k}\right)^{K-k}.$$

Recall $K/4 < k \leq K/4 + 1$. Thus $\left(\frac{K}{4k}\right)^k \leq 1$ and $\left(\frac{3}{4}\frac{K}{K-k}\right)^{K-k} \leq \left(1 + \frac{1}{3K/4-1}\right)^{3K/4} = e + o_K(1)$. We get that $\text{MD}(Y) \leq (1 + o_K(1))\frac{3e}{8}K\sqrt{\frac{K}{2\pi k(K-k)}}$. Recall that $K = \lfloor\frac{3m}{8}\rfloor$ or $K = \lfloor\frac{3m}{8}\rfloor + 1$. In particular, $K = (1 + o_m(1))\frac{8}{3}m$ and $k = (1 + o_m(1))\frac{2}{3}m$. Thus

$$\text{MD}(Y) = (1 + o_m(1))\frac{3e}{8}\frac{8m}{3}\sqrt{\frac{8m/3}{2\pi 2m/3 \cdot (1m/3)}} = (1 + o_m(1))\sqrt{\frac{6e^2m}{\pi}}.$$

Consider any job set $\mathcal{J}$ of size n with $R(\mathcal{J}) \leq \frac{3}{8}$. Combining the previous bound with Lemma 21 and Lemma 22 yields

$$\text{NMD}[L_{1/4}] \leq \frac{8}{3m}\text{MD}[X] = (1 + o_m(1))\frac{8}{3m}\sqrt{\frac{6e^2m}{\pi}} < \frac{10.02 + o_m(1)}{\sqrt{m}}.$$

This bound allows to establish a competitive ratio in the random-order model.

Theorem 24. *The competitive ratio of* LightLoadROM *in the random-order model is* $1.75 + \frac{18}{\sqrt{m}} + O\left(\frac{1}{m}\right)$.

Proof. This is a consequence of Theorem 19 and the prior bound on $\text{NMD}[L_{1/4}]$. □

Remark 1. *The constant in the previous theorem is far from optimal. As mentioned before a first improvement can be derived by estimating the absolute mean average deviation of the hypergeoemtric distribution directly using the techniques from [17]. A much stronger improvement results from filtering out huge jobs reducing essentially to the case that $R(\mathcal{J}) = 1$. Given any guess L_{guess} let $\delta(L_{\text{guess}})$ be $L - L_{\text{guess}}$ if $L_{\text{guess}} \leq L$; $L_{\text{guess}} - \text{OPT}$ if $L_{\text{guess}} > \text{OPT}$; and 0 else. Combining Corollary 12 and Lemma 17 yields that LightLoad$[L_{\text{guess}}]$ is $1.75(1 + \delta(L_{\text{guess}}))$-competitive. So far, we picked for L_{guess} an estimator for L. But besides L we could also try to estimate the lower bound for OPT, that is we could consider $B_{\text{pre}}[\mathcal{J}^\sigma] = \max\left(L_{1/4}[\mathcal{J}^\sigma], p_{\max}^{n/4}[\mathcal{J}^\sigma]\right)$, which is similar to how we estimate OPT in our main algorithm. The nature of the guess $B_{\text{pre}}[\mathcal{J}^\sigma]$ ensures that only few job have size exceeding B_{pre}; only 4 in expectation. A more careful analysis reveals that the competitive ratio of LightLoad$[B_{\text{pre}}]$ is in fact $1.75 + \frac{4.4}{\sqrt{m}} + \frac{7}{m} + O\left(m^{-3/2}\right)$ in the random-order model.*
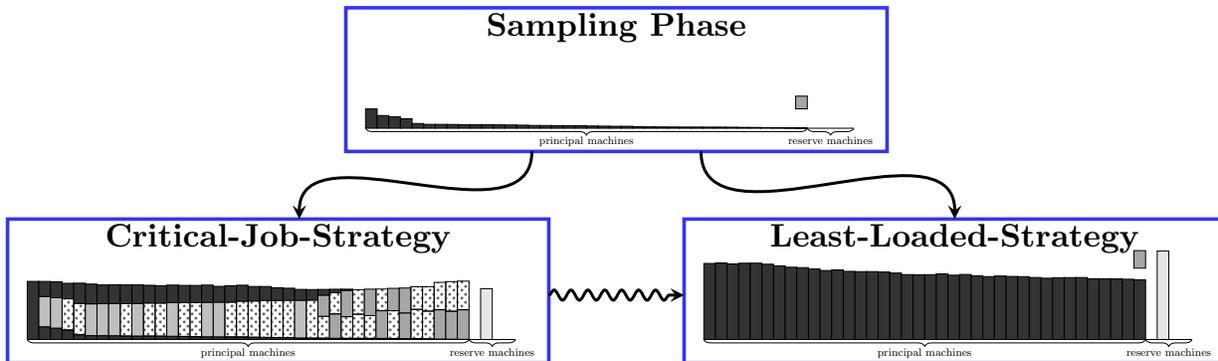
Figure 5: The 1.535-competitive algorithm. First, few jobs are sampled. Then, the algorithm decides between two strategies. The Critical-Job-Strategy tries to schedule critical jobs ahead of time. The Least-Loaded-Strategy follows a greedy approach, which reserves some machines for large jobs. Sometimes, we realize very late that the Critical-Job-Strategy does not work and have to switch to the Least-Loaded-Strategy 'on the fly'. We never switch in the other direction.

5 The new, nearly 1.535-competitive algorithm

Our new algorithm achieves a competitive ratio of $c = \frac{1+\sqrt{13}}{3} \approx 1.535$. Let $\delta = \delta(m) = \frac{1}{\log(m)}$ be the *margin of error our algorithm allows.* Throughout the analysis it is mostly sensible to treat δ as a constant and forget about its dependency on m. Our algorithm maintains a certain set $\mathcal{M}_{\mathrm{res}}$ of $\lceil \delta m \rceil$ *reserve machines.* Their complement, the *principal machines,* are denoted by $\mathcal{M}$. Let us fix an input sequence $\mathcal{J}^\sigma$. Let $\hat{L} = \hat{L}[\mathcal{J}^\sigma] = L_{\delta^2}[\mathcal{J}^\sigma]$. For simplicity, we hide the dependency on $\mathcal{J}^\sigma$ whenever possible. Our online algorithm uses $B = \max\left(p_{\max}^{\delta^2 n}, \hat{L}\right)$ as an *estimated lower bound for* OPT, which is known after the first $\lfloor \delta^2 n \rfloor$ jobs are treated. Our algorithm uses geometric rounding implicitly. Given a job J_t of size p_t let $f(p_t) = (1 + \delta)^{\lfloor \log_{1+\delta} p_t \rfloor}$ be its *rounded size.* We also call J_t an $f(p_t)$-job. Using rounded sizes, we introduce job classes. Let $p_{\mathrm{small}} = c - 1 = \frac{\sqrt{13}-2}{3} \approx 0.535$ and $p_{\mathrm{big}} = \frac{c}{2} = \frac{1+\sqrt{13}}{6} \approx 0.768$. Then we call job J_t

- *small* if $f(p_t) \leq p_{\mathrm{small}}B$ and *critical* else,
- *big* if $f(p_t) > p_{\mathrm{big}}B$,
- *medium* if J is neither small nor big, i.e. $p_{\mathrm{small}}B \leq f(p_t) \leq p_{\mathrm{big}}B$,
- *huge* if its (not-rounded) size exceeds B, i.e. $B < p_t$, and *normal* else.

Consider the sets $\mathcal{P}_{\mathrm{med}} = \{(1 + \delta)^i \mid (1 + \delta)^{-1} p_{\mathrm{small}}B < (1 + \delta)^i \leq p_{\mathrm{big}}B\}$ and $\mathcal{P}_{\mathrm{big}} = \{(1 + \delta)^i \mid p_{\mathrm{big}}B < (1 + \delta)^i \leq B\}$ corresponding to all possible rounded sizes of medium respectively big jobs, excluding huge jobs. Let $\mathcal{P} = \mathcal{P}_{\mathrm{med}} \cup \mathcal{P}_{\mathrm{big}}$. This subdivision gives rise to a *weight function,* which will be important later. Let $w(p) = 1/2$ for $p \in \mathcal{P}_{\mathrm{med}}$ and $w(p) = 1$ for $p \in \mathcal{P}_{\mathrm{big}}$. The elements $p \in \mathcal{P}$ define job classes $\mathcal{C}_p \subseteq \mathcal{J}$ consisting of all p-jobs, i.e. jobs of rounded size p. By some abuse of notation, we call the elements in $\mathcal{P}$ 'job classes', too. Using the notation from Section 4.1 we set $n_p = n_{\mathcal{C}_p} = |\mathcal{C}_p|$ and $\hat{n}_p = n_{\mathcal{C}_p, \delta^2} = |\{J_{\sigma(j)} \mid \sigma(j) \leq \delta^2 n \wedge J_{\sigma(j)} \text{ is a } p\text{-job}\}|$. We want to use the values $\hat{n}_p$, which are available to an online algorithm quite early, to estimate the values n_p, which accurately describe the set of critical jobs. First, $\delta^{-2}\hat{n}_p$ comes to mind as an estimate for

14

n_p. Yet, we need a more complicated guess: $c_p = \max\left(\left\lfloor\left(\delta^{-2}\hat{n}_p - m^{3/4}\right)w(p)\right\rfloor, \hat{n}_p\right)w(p)^{-1}$. It has three desirable advantages. First, for every $p \in \mathcal{P}$ the value c_p is close to n_p with high probability, but, opposed to $\delta^{-2}\hat{n}_p$, unlikely to exceed it. Overestimating n_p turns out to be far worse than underestimating it. Second, $w(p)c_p$ is an integer and, third, we have $c_p \geq \hat{n}_p w(p)^{-1}$. A fundamental fact regarding the values $(c_p)_{p\in\mathcal{P}}$ and B is, of course, that they are known to the online algorithm once $\lfloor\delta^2 n\rfloor$ jobs are scheduled.

Statement of the algorithm: If there are less jobs than machines, i.e. $n \leq m$, it is optimal to put each job onto a separate machine. Else, a short sampling phase greedily schedules each of the first $\lfloor\delta^2 n\rfloor$ jobs to the least loaded principal machine $M \in \mathcal{M}$. Now, the values B and $(c_p)_{p\in\mathcal{P}}$ are known. Our algorithm has to choose between two strategies, the Least-Loaded-Strategy and the Critical-Job-Strategy, which we will both introduce subsequently. It maintains a variable STRAT, initialized to CRITICAL, to remember its choice. If it chooses the Critical-Job-Strategy, some additional preparation is required. It may at any time discover that the Critical-Job-Strategy is not feasible and switch to the Least-Loaded-Strategy but it never switches the other way around.

Algorithm 2 The complete algorithm: How to schedule job J_t.
___
1: STRAT *is initialized to* CRITICAL*, J_t is the job to be scheduled.*
2: **if** $n \leq m$ **then** Schedule J_t on any empty machine;
3: **else if** $t \leq \varphi n$ **then** schedule J_t on a least loaded machine in $\mathcal{M}$;   ▷ *Sampling phase*
4: **else**
5:    **if** we have $t = \lfloor\varphi n\rfloor + 1$ **then**
6:       **if** $\sum_{p\in\mathcal{P}} w(p)c_p > m$ **then**   STRAT $\leftarrow$ LEAST-LOADED
7:       **else** proceed with the Preparation for the Critical-Job-Strategy (Algorithm 4);
8:    **if** STRAT = CRITICAL **then** proceed with the Critical-Job-Strategy (Algorithm 5);
9:    **else** proceed with the Least-Loaded-Strategy (Algorithm 3);
___

The **Least-Loaded-Strategy** places any normal job on a least loaded principal machine. Huge jobs are scheduled on any least loaded reserve machine. This machine will be empty, unless we consider rare worst-case orders.
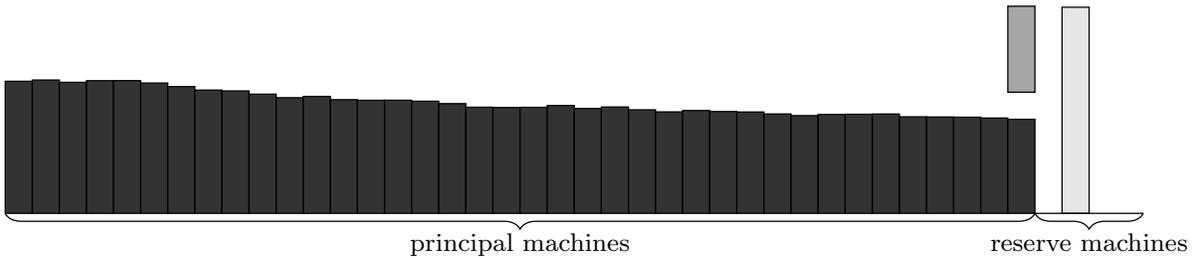

Figure 6: The Least-Loaded-Strategy schedules jobs greedily. A few machines are reserved for unexpected huge jobs, such as the largest job, which is unlikely to arrive in the sampling phase.

Algorithm 3 The Least-Loaded-Strategy: How to schedule job J_t.

1: **if** J_t is huge **then** schedule J_t on any least loaded reserve machine;
2: **else** schedule J_t on any least loaded principal machine;

For the Critical-Job-Strategy we introduce *p-placeholder-jobs* for every size $p \in \mathcal{P}$. Sensibly, the size of a p-placeholder-job is p. During the Critical-Job-Strategy we treat placeholder-jobs similar to *real* jobs. The *anticipated load* $\tilde{l}_M^t$ of a machine M at time t is the sum of all jobs on it, including placeholder-job, opposed to the common load l_M^t, which does not take the latter into account. Note that $\tilde{l}_M^t$ defines a pseudo-load as introduced in Section 4.
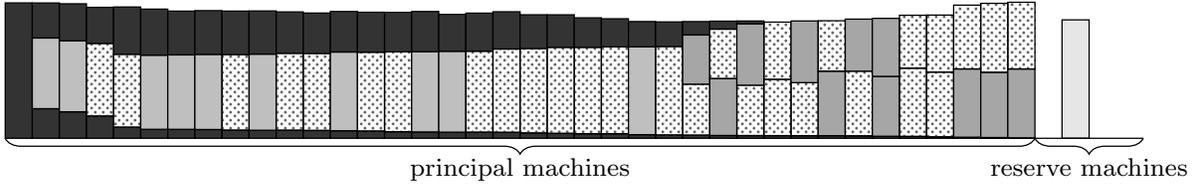

Figure 7: The Critical-Job-Strategy. Each machine gets either two medium, one large or no critical job. Placeholder jobs (dotted) are assigned during the Preparation and reserve space for critical jobs yet to come. Processing volume of small jobs (dark) 'on the bottom' arrived during the sampling phase. Reserve machines accommodate huge jobs or, possibly, jobs without matching placeholders.

During the **Preparation for the Critical-Job-Strategy** the algorithm maintains a counter c_p' of all p-jobs scheduled so far (including placeholders). A job class $p \in \mathcal{P}$ is called *unsaturated* if $c_p' \leq c_p$. First, we add unsaturated medium placeholder-jobs to any principal machine that already contains a medium real job from the sampling phase. We will see in Lemma 25 that such an unsaturated medium job class always exists. Now, let m_{empty} be the number of principal machines which do not contain critical jobs. We prepare a set $\mathcal{J}_{\text{rep}}$ of cardinality at most m_{empty}, which we will then schedule onto these machines. The set $\mathcal{J}_{\text{rep}}$ may contain single big placeholder-jobs or pairs of medium placeholder-jobs. We greedily pick any unsaturated job class $p \in \mathcal{P}$ and add a p-placeholder-job to $\mathcal{J}_{\text{rep}}$. If p is medium, we pair it with a job belonging to any other, not necessarily different, unsaturated medium job class. Such a job class always exists by Lemma 25. We stop once all job classes are saturated or if $|\mathcal{J}_{\text{rep}}| = m_{\text{empty}}$. We then assign the elements in $\mathcal{J}_{\text{rep}}$ to machines. We iteratively pick the element $e \in \mathcal{J}_{\text{rep}}$ of maximum size and assign the corresponding jobs to the least loaded principal machine, which does not contain critical jobs yet. Sensibly, the size of a pair of jobs in $\mathcal{J}_{\text{rep}}$ is the sum of their individual sizes. We repeat this until all jobs and job pairs in $\mathcal{J}_{\text{rep}}$ are assigned to some principal machine.

16

Algorithm 4 Preparation for the Critical-Job-Strategy.

1: **while** there is a machine M containing a single medium job **do**
2:     Add a placeholder p-job for an unsaturated size class $p \in \mathcal{P}_{\mathrm{med}}$ to M; $c'_p \leftarrow c'_p + 1$;

3: **while** there is an unsaturated size class $p \in \mathcal{P}$ and $|\mathcal{J}_{\mathrm{rep}}| < m_{\mathrm{empty}}$ **do**
4:     Pick an unsaturated size class $e = p \in \mathcal{P}$ with c'_p minimal; $w(e) \leftarrow p$; $c'_p \leftarrow c'_p + 1$;
5:     **if** p is medium **then** pick $q \in \mathcal{P}_{\mathrm{med}}$ unsaturated. $e \leftarrow (p, q)$; $w(e) \leftarrow p + q$; $c'_q \leftarrow c'_q + 1$;

6:     Add e to $\mathcal{J}_{\mathrm{rep}}$;

7: **while** $\mathcal{J}_{\mathrm{rep}} \neq \emptyset$ **do**
8:     Pick a least loaded machine $M \in \mathcal{M}$, which does not contain a critical job yet;
9:     Pick $e \in \mathcal{J}_{\mathrm{rep}}$ of maximum size $w(e)$ and add the jobs in e to M;
10:     $\mathcal{J}_{\mathrm{rep}} \leftarrow \mathcal{J}_{\mathrm{rep}} \setminus \{e\}$;

Lemma 25. *In line 2 and 5 of Algorithm 4 there is always an unsaturated medium size class available. Thus, Algorithm 4, the Preparation for the Critical-Job-Strategy, is well defined.*

Proof. Concerning line 2, there are precisely $\sum_{p \in \mathcal{P}_{\mathrm{med}}} \hat{n}_p$ machines with critical jobs while there are at least $\sum_{p \in \mathcal{P}_{\mathrm{med}}} (c_p - \hat{n}_p) \geq \sum_{p \in \mathcal{P}_{\mathrm{med}}} \hat{n}_p$ placeholder-jobs available to fill them. Here we make use of the fact that for medium jobs $p \in \mathcal{P}_{\mathrm{med}}$ we have $c_p \geq \hat{n}_p w(p)^{-1} = 2\hat{n}_p$.

Concerning line 5, observe that so far every machine and every element in $\mathcal{J}_{\mathrm{rep}}$ contains an even number of medium jobs. If the placeholder picked in line 4 was the last medium job remaining, $\sum_{p \in \mathcal{P}_{\mathrm{med}}} c_p$ would be odd. But this is not the case since every c_p for $p \in \mathcal{P}_{\mathrm{med}}$ is even. $\qquad\square$

After the Preparation is done, the **Critical-Job-Strategy** becomes straightforward. Each small job is scheduled on a principal machines with least anticipated load, i.e. taking placeholders into account. Critical jobs of rounded size $p \in \mathcal{P}$ replace p-placeholder-jobs whenever possible. If no such placeholder exists anymore, critical jobs are placed onto the reserve machines. Again, we try pair up medium jobs whenever possible. If no suitable machine can be found among the reserve machines, we have to switch to the Least-Loaded-Strategy. We say that the algorithm *fails* if it ever reaches this point. In this case, it should rather have chosen the Least-Loaded-Strategy to begin with. Since all reserve machines are filled at this point, the Least-Loaded-Strategy is impeded, too. The most difficult part of our analysis shows that, excluding worst-case orders, this is not a problem on job sets that are prone to cause failing.

Algorithm 5 The Critical-Job-Strategy.

1: **if** J_t is medium or big **then** *let p denote its rounded size;*
2:     **if** there is a machine M containing a p-placeholder-job J **then**
3:         Delete the p-placeholder-job J and assign J_t to M;
4:     **else if** J_t is medium and there exists $M \in \mathcal{M}_{\mathrm{res}}$ containing a single medium job **then**
5:         Schedule J_t on M;
6:     **else if** there exists an empty machine $M \in \mathcal{M}_{\mathrm{res}}$ **then** schedule J_t on M;
7:     **else** STAT $\leftarrow$ LEAST-LOADED;   ▷ We say the algorithm *fails.*
8:         **use** the Least-Loaded-Strategy (Algorithm 3) from now on;
9: **else** assign J_t to the least loaded machine in $\mathcal{M}$ (take placeholder jobs into account);

---

6 Analysis of the algorithm

Theorem 26 is main result of the paper. Corollary 27 follows immediately by Lemma 2.

Theorem 26. *Our algorithm is nearly c-competitive. Recall that $c = \frac{1+\sqrt{13}}{3} \approx 1.535$.*

Corollary 27. *Our algorithm is c-competitive in the secretary model as $m \to \infty$.*

The analysis of our algorithm proceeds along the same three reduction steps used in the proof of Theorem 15. First, we assert that our algorithm has a bounded adversarial competitive ratio, which approaches 1 as $R(\mathcal{J}) \to 0$. Not only does this lead to the first condition of nearly competitiveness, it also enables us to introduce *simple* job sets on which we perform well due to basic considerations resulting from Section 4.

Definition 28. *A job set $\mathcal{J}$ is called* simple *if $R(\mathcal{J}) \leq \frac{(1-\delta)\delta^3}{2(\delta^2+1)}(2-c)$ or if it consists of at most m jobs. Else, we call it* proper. *We call any ordered input sequence $\mathcal{J}^\sigma$* simple *respectively* proper *if the underlying set $\mathcal{J}$ has this property.*

Next we are going to sketch out main proof introducing three Main Lemmas. These follow the three proof steps introduced in the proof of Corollary 16.

Main Lemma 1. *In the adversarial model our algorithm has competitive ratio $4 + O(\delta)$ on general input sequences and $c + O(\delta)$ on simple sequences.*

The proof is discussed later. We are thus reduced to treating *proper* job sets. In the second reduction we introduce *stable* sequences. These have many desirable properties. Most notably, they are suited to sampling. We leave the formal definition to Section 6.2 since it is rather technical. The second reduction shows that stable sequences arise with high probability if one orders a proper job set uniformly randomly.

Formally, for m the number of machines, let $P(m)$ be the maximum probability by which the permutation of any proper sequence may not be stable, i.e.

$$P(m) = \sup_{\mathcal{J} \text{ proper}} \mathbf{P}_{\sigma \sim S_n} \left[\mathcal{J}^\sigma \text{ is not stable} \right].$$

The second main lemma asserts that this probability vanishes as $m \to \infty$.

Main Lemma 2. $\lim_{m \to \infty} P(m) = 0.$

In other words, non-stable sequences are very rare and of negligible impact in random-order analyses. Thus, we only need to consider stable sequences. In the final, third, step we analyze our algorithm on these. This analysis is quite general. In particular, it does not rely further on the random-order model. Instead, we work with worst-case stable input sequences, i.e. we allow the adversary to present any (ordered) stable input sequence.

Main Lemma 3. *Our algorithm is adversarially $(c + O(\delta))$-competitive on stable sequences.*

These three main lemmas allow us to conclude the proof of Theorem 26.

Proof of Theorem 26. By Main Lemma 1, the first condition of nearly competitiveness holds, i.e. our algorithm has a constant competitive ratio. Moreover, by Main Lemma 1 and Main Lemma 3, given $\varepsilon > 0$, we can pick $m_0(\varepsilon)$ such that our algorithm is $(c + \varepsilon)$-competitive on all sequences that are stable or simple if there are at least $m_0(\varepsilon)$ machines. Here, we need that $\delta(m) \to 0$ for $m \to \infty$. This implies that for $m \geq m_0(\varepsilon)$ the probability of our algorithm not being $(c + \varepsilon)$-competitive is at most $P(m)$, the maximum probability with which a random permutation of a proper, i.e. non-simple, input sequence is not stable. By Main Lemma 2, we can find $m(\varepsilon) \geq m_0(\varepsilon)$ such that this probability is less than ε. This satisfies the second condition of nearly competitiveness. □

6.1 The adversarial case. Proof of Main Lemma 1

Recall that the *anticipated load* $\tilde{l}_M^t$ of a machine M at time t denotes its load including placeholder-jobs. It satisfies the definition of a pseudo-load as introduced in Section 4. We obtain the following two bounds on the average anticipated load $\tilde{L} = \sup_t \frac{1}{m} \sum_M \tilde{l}_M^t$.

Lemma 29. *We have $\tilde{L} \leq L + 2p_{\max}$. In particular $\tilde{R}(\mathcal{J}) \leq 3$.*

Proof. First note that every placeholder-job has at most the size of some job encountered during the sampling phase. In particular, the size of any placeholder-job is at most $p_{\max}$. Since there are at most two placeholder-jobs on each machine, the total processing time of all placeholder-jobs is at most $2mp_{\max}$. The total processing time of real jobs is at most mL. Thus the total processing time of all placeholder and real jobs scheduled at any time cannot exceed $m(L + 2p_{\max})$. In particular, at any time t, we have $\frac{1}{m} \sum_M \tilde{l}_M^t \leq \frac{1}{m} m(L + 2p_{\max})$. Thus, $\tilde{L} \leq L + 2p_{\max}$. For the second part we conclude that $\tilde{R}(\mathcal{J}) \leq \min\left(\frac{\tilde{L}}{p_{\max}}, \frac{\tilde{L}}{L}\right) \leq \min\left(\frac{L}{p_{\max}} + 2, 2\frac{p_{\max}}{L} + 1\right) \leq 3$. □

Lemma 30. *We have $\tilde{L} \leq \left(1 + \frac{1}{\delta^2}\right) L$, in particular $\tilde{R}(\mathcal{J}) \leq \left(1 + \frac{1}{\delta^2}\right) R(\mathcal{J})$.*

Let us first show the following stronger lemma.

Lemma 31. *The total size of the placeholder-jobs is at most $m\hat{L}$. In particular, $\tilde{L} \leq L + \hat{L}$.*

Proof of Lemma 31. For every $p \in \mathcal{P}$ we schedule at most $c_p \leq \delta^{-2}\hat{n}_p$ placeholder-jobs of type p. Thus, the total size of the placeholder-jobs is at most $\sum_{p \in \mathcal{P}} \delta^{-2}\hat{n}_p \leq m\hat{L}$. The total size of all real jobs is precisely mL. Since $\tilde{L}$ is at most $\frac{1}{m}$-times the total processing time of all jobs, we have $\tilde{L} \leq \frac{1}{m}(mL + m\hat{L})$. □

Proof of Lemma 30. Observe that $\hat{L} = L_{\delta^2} \leq \delta^{-2}L$. Then, the bound follows from Lemma 31. □

We call a machine *critical* if it receives a critical job from the Critical-Job-Strategy but no small job after the sampling phase. Else, we call it *general*. General machines can be analyzed using Proposition 5 and 8. *Critical* machines need more careful arguments.

Lemma 32. *At any time, the load of any general machine is at most $\left(\frac{\tilde{R}(\mathcal{J})}{1-\delta} + 1 + 2\delta\right) \text{OPT}(\mathcal{J})$.*

Proof. For sequences of length $n \leq m$ our algorithm is optimal. Hence assume $n > m$.

During the sampling phase and the Least-Loaded-Strategy, our algorithm always uses either a least loaded machine or a least loaded principal machine. Both lie among the $\lfloor \delta m \rfloor + 1$ least loaded machines. By Proposition 5 this cannot cause any load to exceed $\left(\frac{m}{m - \lfloor \delta m \rfloor} R(\mathcal{J}) + 1\right) \text{OPT}(\mathcal{J}) \leq$

$\left(1 + \frac{R(\mathcal{J})}{1-\delta}\right) \mathrm{OPT}(\mathcal{J})$. Observing that $\tilde{R}(\mathcal{J}) \geq R(\mathcal{J})$, see Lemma 6, the lemma holds for every machine that does not receive its last job during the Critical-Job-Strategy.

Now consider a general machine M, which received its last job during the Critical-Job-Strategy. Since it is a general machine, it also received a small job during the Critical-Job-Strategy. Let J be the last small job it received. Right before receiving J machine M must have been a principal machine of least anticipated load. In total, it had at most the $(\lfloor \delta m \rfloor + 1)$-smallest anticipated load. By Proposition 8 its anticipated load was at most $\left(\frac{\tilde{R}(\mathcal{J})}{1-\delta} + 1\right) \mathrm{OPT}(\mathcal{J})$ after receiving J. Afterwards machine M may have received up to two critical jobs, which replaced placeholder-jobs. Since these jobs had at most $(1+\delta)$-times the size of the job they replaced. The load-increase is at most $\delta p_{\max} \leq \delta \mathrm{OPT}$ for each of these two jobs. $\qquad \square$

We can now consider critical machines.

Lemma 33. *The load of a reserve machine is at most $\min(\max((1+\delta)cB, p_{\max}), 2p_{\max})$ till it receives a job from the Least-Loaded-Strategy. Critical reserve machines in particular fulfill this condition.*

Proof. Every critical reserve machine receives either one big job or at most two medium ones, until the Least-Loaded-Strategy is applied. The second bound, $2p_{\max}$, follows immediately from that. The first bound follows from the fact that a single big job has size at most $p_{\max}$, while two medium jobs have size at most $2(1+\delta)p_{\mathrm{big}}B = (1+\delta)cB$. The $(1+\delta)$-factor comes from using rounded sizes in the definition of medium jobs. $\qquad \square$

The following lemma uses similar arguments to Lemma 46 only for the adversarial model.

Lemma 34. *The load of a critical machine is at most $\min((1+\delta)cB + 2\frac{\tilde{R}(\mathcal{J})}{(1-\delta)}\mathrm{OPT}, \frac{L}{1-\delta} + 3p_{\max})$ if it was a principal machine.*

Proof. Consider any critical principal machine M. Let J be the last job received in the sampling phase. Before J was scheduled on M it was a least loaded principle machine and thus had load at most $\frac{L}{1-\delta}$ by Proposition 5. After J machine M received at most two more jobs and thus its load cannot exceed $\frac{L}{1-\delta} + 3p_{\max}$, the second term in the min-term.

If J was critical, this implies that the load on M of non-critical jobs was at most $\frac{L}{1-\delta}$, while the load of critical jobs no M cannot exceed $2(1+\delta)p_{\mathrm{big}}B = (1+\delta)cB$. The first term in the min-term follows. We are left to consider the case that M did not receive a critical job in the sampling phase, which means that it receives an element of $\mathcal{J}_{\mathrm{rep}}$, else M would not be critical. In fact, assume that M was the i-th machine to receive an element from $\mathcal{J}_{\mathrm{rep}}$.

First consider the case $i \leq m/2 - 1$. Right before the while loop in line 7 of Algorithm 4 machine M had the i-th least anticipated load among the principal machines. By Lemma 7 its anticipated load was at most $\frac{m}{m-i-\delta m+1}\tilde{L} \leq \frac{2\tilde{L}}{1-\delta} \leq 2\frac{\tilde{R}(\mathcal{J})}{1-\delta}\mathrm{OPT}$ before receiving placeholder jobs of processing volume at most cB. The processing volume of the placeholder jobs increases by at most a factor $(1+\delta)$ once they are replaced by real jobs. Thus the bound of the lemma follows if $i \leq m/2 - 1$.

Finally, consider the case $i \geq m/2$. Recall that $\delta^{-2}L_{\delta^2} = \hat{L} \leq B$. Since M did not receive a critical job in the sampling phase it follows from Lemma 4 that its load was at most $\frac{\delta^{-2}L_{\delta^2}}{1-\delta} + (c-1)B \leq (c - \frac{\delta}{1-\delta})B \leq (1+\delta) \cdot cB$ after the sampling phase. Let p be the processing volume machine M

receives from $\mathcal{J}_{\mathrm{rep}}$. Since the algorithm assigns the elements of $\mathcal{J}_{\mathrm{rep}}$ in decreasing order at least i machines received processing volume at least p from $\mathcal{J}_{\mathrm{rep}}$. Thus $i \cdot p \leq m \cdot \tilde{L}$ and, using that $i \geq m/2$, we derive that $p \leq \frac{m}{i}\tilde{L} \leq 2\tilde{L} \leq 2\frac{\tilde{R}(\mathcal{J})}{1-\delta}\mathrm{OPT}$. Again the first term of the min-term follows. $\qquad\square$

From these lemmas the two statements of Main Lemma 1 follow.

Corollary 35. *Our algorithm is adversarially* $\left(3 + \frac{3}{1-\delta} + 2\delta\right)$*-competitive.*

Proof. By Lemma 29 we have $\tilde{R}(\mathcal{J}) \leq 3$, also recall that $L, p_{\max} \leq \mathrm{OPT}$. By Lemma 32, 33 and 34 the makespan of the algorithm is thus at most

$$\max\left(\frac{3}{1-\delta} + 1 + 2\delta, 2, \frac{1}{1-\delta} + 3\right)\mathrm{OPT}(\mathcal{J}) = \left(1 + \frac{3}{1-\delta} + 2\delta\right)\mathrm{OPT}(\mathcal{J}). \qquad\square$$

Corollary 36. *Our algorithm has makespan at most* $(c + 2\delta)\mathrm{OPT}$ *on simple sequences* $\mathcal{J}^{\sigma}$.

Proof. By Lemma 32, 33 and 34 we see that the makespan of our algorithm is at most

$$\max\left(\left(\frac{\tilde{R}(\mathcal{J})}{1-\delta} + 1 + 2\delta\right)\mathrm{OPT}(\mathcal{J}), p_{\max}, (1+\delta)cB + 2\frac{R(\mathcal{J})}{1-\delta}\right).$$

Now, by lemma Lemma 30 and the definition of simple sequences, there holds $\tilde{R}(\mathcal{J}) \leq \left(1 + \frac{1}{\delta^2}\right)R(\mathcal{J}) \leq (1-\delta)\frac{\delta}{2}(2-c)$. In particular, $\left(\frac{\tilde{R}(\mathcal{J})}{1-\delta} + 1 + 2\delta\right)\mathrm{OPT}(\mathcal{J}) \leq (c + 2\delta)\mathrm{OPT}(\mathcal{J})$. The second term $p_{\max}$ is always smaller than OPT. Concerning the third bound in the max-term observe using Lemma 31 that there holds

$$\begin{aligned}
B &= \max\left(p_{\max}^{\delta^2 n}, \hat{L}\right) \\
&\leq \max\left(p_{\max}, \delta^{-2}L\right) \\
&\leq \max\left(p_{\max}, \delta^{-2}R(\mathcal{J})p_{\max}\right) \\
&\leq \max\left(p_{\max}, \delta^{-2}\frac{(1-\delta)\delta^3}{2(\delta^2+1)}(2-c)p_{\max}\right) \\
&\leq p_{\max} \\
&\leq \mathrm{OPT}.
\end{aligned}$$

Since $2\frac{\tilde{R}(\mathcal{J})}{1-\delta} \leq (2-c)\delta$ we have $(1+\delta)cB + 2\frac{R(\mathcal{J})}{1-\delta}\mathrm{OPT} \leq (c+2\delta)\mathrm{OPT}$. $\qquad\square$

Proof of Main Lemma 1. Main Lemma 1 follows immediately from Corollary 35 and Corollary 36. $\qquad\square$

6.2 Stable job sequences. Proof sketch of Main Lemma 2

We introduce the class of *stable* job sequences. The first two conditions state that all estimates our algorithm makes are *accurate*, i.e. sampling works. By the third condition there are less huge jobs than reserve machines and the fourth condition states that these jobs are distributed evenly. The final condition is a technicality. Stable sequences are useful since they occur with high probability if we randomly order a proper job set.

Definition 37. *A job sequence $\mathcal{J}^\sigma$ is* stable *if the following conditions hold:*

- *The estimate $\hat{L}$ for L is accurate, i.e. $(1-\delta)L \leq \hat{L} \leq (1+\delta)L$.*
- *The estimate c_p for n_p is accurate, i.e. $c_p \leq n_p \leq c_p + 2m^{3/4}$ for all $p \in \mathcal{P}$.*
- *There are at most $\lceil \delta m \rceil$ huge jobs in $\mathcal{J}^\sigma$.*
- *Let $\tilde{t}$ be the time the last huge job arrived and let $n_{p,\tilde{t}}$ be the number of p-jobs scheduled at that time for a given $p \in \mathcal{P}$. Then $n_{p,\tilde{t}} \leq \left(1-\delta^3\right)n_p$ for every $p \in \mathcal{P}$ with $n_p > \left\lfloor \frac{(1-\delta-2\delta^2)m}{|\mathcal{P}|} \right\rfloor$.*
- $\delta^3 \left\lfloor \left(1-\delta-2\delta^2\right)m/|\mathcal{P}| \right\rfloor \geq 2|\mathcal{P}|m^{3/4}$.

Proof sketch of Main Lemma 2. The first two conditions are covered by arguments following Section 4.1. Here, we require that only proper sequences are considered. The third condition is equivalent to demanding one of the $\lceil \delta m \rceil$ largest jobs to occur during the sampling phase. This is extremely likely. The expected rank of the largest job occurring in the sampling phase is δ^{-2}, a constant. The fourth condition states that, for any $p \in \mathcal{P}$, the huge jobs are evenly spread throughout the sequence when compared to any sizable class of p-jobs. Again, this is expected of a random sequence and corresponds to how one would view randomness statistically. For the final condition it suffices to choose the number of machines m large enough. One technical problem arises since the class $\mathcal{P} = \mathcal{P}[\mathcal{J}^\sigma]$ is defined using the value $B[\mathcal{J}^\sigma]$. It thus highly depends on the input permutation σ. We rectify this by passing over to a larger class $\hat{\mathcal{P}}$ such that $\mathcal{P} \subset \hat{\mathcal{P}}$ with high probability. $\qquad\square$

The formal proof of Main Lemma 2 is simple but very technical. That is, we consider the underlying ideas to be rather simple but in order to give a rigorous proof many cases have to be considered. We leave it to Appendix C. The definition of stable sequences is suited for our future algorithmic arguments. To make probabilistic arguments, we introduce *probabilistically stable* sequences and prove that probabilistically stable sequences are always stable. Their definition is more convenient, as it avoids certain problems such as $\mathcal{P}$ being dependent on the job permutation. We then prove Main Lemma 2 for all six conditions of probabilistically stable sequences separately.

6.3 Adversarial analysis on stable sequences. Proof sketch of Main Lemma 3

General observations

This section is devoted for some general observations needed several times throughout the analysis. Recall that $\tilde{L} = \sup_t \frac{1}{m} \sum_M \tilde{l}_M^t$ denotes the maximum average load taking placeholder jobs into account. We will see that this does, in fact, not overestimate the total load L if the sequence is stable.

Lemma 38. *For every stable sequence $\mathcal{J}^\sigma$ there holds $\tilde{L} = L$.*

Proof. By Lemma 6 we have $\tilde{L} \geq L$ for any pseudo-load. Recall that $\tilde{L} = \sup_t \frac{1}{m} \sum_M \tilde{l}_M^t$. Thus it suffices to show that $\sup_t \frac{1}{m} \sum_M \tilde{l}_M^t \leq L$ for any given time t. Consider the schedule of our algorithm time t including placeholder-jobs. If it contains p-placeholder-jobs for some $p \in \mathcal{P}$ it contains at most c_p many p-jobs in total. By the second property of stable sequences there holds $c_p \leq n_p$. Thus, we can find real p-jobs not scheduled yet and replace the p-placeholder-jobs by them. This way the load of each machine can only increase. In particular, the resulting schedule has average load at least $\frac{1}{m} \sum_M \tilde{l}_M^t$. But since it contains only real jobs, its average load will be at most L. Therefore $\frac{1}{m} \sum_M \tilde{l}_M^t \leq L$. $\qquad\square$

The following lemma is a basic but very useful observation describing the load of any machine after the sampling phase.

Lemma 39. *Let M be any machine after the sampling phase and p be the size of the largest job scheduled on it. Then the load of M is at most $\frac{\delta^2}{1-\delta}B + p$.*

Proof. Let l be the load of M before the last job J was scheduled on it. Using Lemma 4 we see that $l \le \frac{m}{m-\lfloor\delta m\rfloor}\delta^2\hat{L} \le \frac{\delta^2}{1-\delta}B$. The last inequality uses $L_{\delta^2} = \hat{L} \le B$. Since J had size at most p the lemma follows. $\square$

Lemma 40. *Till the Least-Loaded-Strategy is employed (or till the end of the sequence) there is at most one reserve machine M whose only critical job is medium. Every other machine contains either no critical job, one big job or two medium jobs (including placeholder jobs).*

Proof. First consider the situation right before the Critical-Job-Strategy is employed. Let M be a machine containing a critical job. By Lemma 39 the total size of all jobs besides the largest one on M is at most $\delta^2\hat{L} \le \delta^2 B$. Since this is less than $p_{\text{small}}B$ only the largest job could have been critical. Now observe that the algorithm adds a second medium placeholder-job to precisely every machine that contained a (necessarily single) medium job after the sampling phase. Afterwards, medium placeholder-jobs are always scheduled in pairs onto machines which do not contain critical jobs. While the Critical-Job-Strategy is employed, the number of medium jobs does not change for principal machines. We only replace placeholders with real jobs. Moreover the algorithm ensures that at most one reserve machine M has a single medium job. $\square$

Finally let us make the following technical observation, which will be necessary later.

Lemma 41. *There are at most $2\delta^2 m$ machines which contain (real) critical jobs before the Preparation for the Critical-Job-Strategy. In particular $m_{\text{empty}} \le \left(1 - \delta - 2\delta^2\right)m$.*

Proof. Assume the lemma would not hold. Since each critical job has size at least $p_{\text{small}}B$ this implies that $B \ge \hat{L} > \frac{1}{\delta^2 m} \cdot 2\delta^2 m \cdot p_{\text{small}}B = 2p_{\text{small}}B > B$. A contradiction. In particular, at most $2\delta^2 m$ machines received critical jobs after the observational phase. Thus $m_{\text{empty}} \le |\mathcal{M}| - 2\delta^2 m \le (1 - \delta - 2\delta^2)m$. $\square$

Before the Least-Loaded-Strategy is employed.

The goal of this section is to analyze every part of the algorithm but the Least-Loaded-Strategy. Formally we want to show the following proposition and its important Corollary 43.

Proposition 42. *The makespan of our algorithm is at most $(c + O(\delta)) \max{(B, L, p_{\text{max}})}$ on stable sequences till it employs the Least-Loaded-Strategy (or till the end of the sequence).*

For a formal proof we need to consider many cases where the statement of the lemma could go wrong. Let us first give a sketch of the full proof, which will be fleshed out subsequently.

Proof sketch. Let us only consider critical jobs at any time the Least-Loaded-Strategy, Algorithm 3, is not employed. Our algorithm ensures that a machine contains either one big job or at most two medium jobs. Formally, this is shown in Lemma 40. In the first case, we simply bound the size of this big, possibly huge, job by p_{max}. Else, if the machine contains up to two medium jobs their total

size is at most $2(1+\delta)p_{\text{big}}B = (1+\delta)cB$. The factor $(1+\delta)$ arises since we use rounded sizes in the definition of medium jobs. Thus, critical jobs may cause a load of at most $\max(p_{\max}, (c+O(\delta))B)$.

Analyzing the load increase by small, i.e. non-critical, jobs relies on Proposition 5 and 8 depending on whether these jobs were assigned during the sampling phase or during the Critical-Job-Strategy. □

Note that for stable sequences $\hat{L} \le (1+\delta)L \le (1+\delta)\text{OPT}$, in particular $\max(B, L, p_{\max}) = \max\left(p_{\max}^{\delta^2 n}, \hat{L}, L, p_{\max}\right) \le (1+\delta)\text{OPT}$. This proves the following important corollary to Proposition 42.

Corollary 43. *Till the Least-Loaded-Strategy is used the makespan of our algorithm is at most $(c+O(\delta))\text{OPT}$ on stable sequences.*

We first need to assert that the statement holds after the preparation for the Critical-Job-Strategy, namely we prove the following proposition.

Proposition 44. *After the Preparation for the Critical-Job-Strategy the anticipated load of no machine exceeds $(c+O(\delta))B$.*

There are three types of machines we need to consider. First, there are machines which contain a real critical job after the Preparation for the Critical Job Strategy. Second, there are machines, which only receive placeholder jobs. Finally there are machines that only receive critical jobs during sampling. The following two lemmas concern themselves with the first two types of machines. Afterwards we prove Proposition 44.

Lemma 45. *If a machine contains a real critical job its anticipated load is at most $((1+\delta/2)c + \frac{\delta^2}{1-\delta})B \le (c+O(\delta))B$ after the Preparation for the Critical-Job-Strategy.*

Proof. After the Preparation for the Critical-Job-Strategy a machine contains either a big job of size at most $B < (1+\delta/2)cB$ or two medium jobs. Each medium has size at most $(1+\delta)p_{\text{big}}$ where the factor $(1+\delta)$ is due to rounding. Thus the total size of critical jobs is at most $2(1+\delta)p_{\text{big}}B = (1+\delta/2)cB$. Lemma 39 bounds the size of all non-critical jobs by $\frac{\delta^2}{1-\delta}B$. □

Lemma 46. *Let M be the i-th last machine that received a job from $\mathcal{J}_{\text{rep}}$ for $i \le m_{\text{empty}}$. After the Preparation for the Critical-Job-Strategy its anticipated load is at most*

$$\min\left(p_{\text{small}} + \frac{\delta^2}{1-\delta}, \frac{m}{i}\delta^2\right)B + \min\left(c, \frac{m}{m_{\text{empty}} - i + 1}\right)B \le (c+O(\delta))B.$$

Proof. Let l be the load of M before the Preparation for the Critical-Job-Strategy and let p be the sum of all the placeholder-jobs assigned to it. Then the load of M after the preparation is precisely $l + p$. We bound both summands separately

To see that $l \le \min\left(p_{\text{small}} + \frac{\delta^2}{1-\delta}, \frac{m}{i}\delta^2\right)B$ observe that the largest job on M has size at most $p_{\text{small}}B$ since M does contain no critical jobs. In particular, by Lemma 39, $l \le (p_{\text{small}} + \frac{\delta^2}{1-\delta})B$. Consider the schedule right before placeholder jobs were assigned. By definition this schedule had average load $\frac{\delta^2}{1-\delta}\hat{L} \le \frac{\delta^2}{1-\delta}B$ and M was at most the i-th most loaded machine. The second bound then follows from Lemma 4.

We have $p \le \min\left(c, \frac{m}{m_{\text{empty}}-i+1}\right)B$. The first bound holds since we either assign two medium placeholder-jobs of size at most $p_{\text{big}}B$ each or one big job of size at most B to any machine. Thus the sum of the placeholder-jobs assigned is at most $\max(1, 2p_{\text{big}})B = cB$. For the second term recall that Lemma 31 shows that the total size of all placeholder-jobs is at most $m\hat{L} \le mB$. Prior to M precisely $m_{\text{empty}} - i$ machines received placeholder job of total size at least p. Thus, $(m_{\text{empty}} - i + 1)p \le mB$, or, equivalently, $p \le \frac{m}{m_{\text{empty}}-i+1}B$.

Altogether we derive that the anticipated load of M is $l + p \le \min\left(p_{\text{small}} + \frac{\delta^2}{1-\delta}, \frac{m}{i}\delta^2\right)B + \min\left(c, \frac{m}{m_{\text{empty}}-i+1}\right)B$. We need to see that this term is in $(c + O(\delta))B$. Consider two cases. For $i \ge \delta m$ the term is at most $\frac{m}{i}\frac{\delta^2}{1-\delta}B + cB \le (c+\delta)B$. Else, for $i \le \delta m$, it is at most $(p_{\text{small}}+\delta^2)B+\frac{m}{m_{\text{empty}}-i+1}B \le \left(p_{\text{small}} + \delta^2 + \frac{m}{m-2\delta m-\delta^2 m}\right)B = \left(c+\delta^2 + \frac{2\delta-\delta^2}{1-2\delta+\delta^2}\right)B = (c+O(\delta))B$. The first inequality uses Lemma 41, the second equality uses that $p_{\text{small}} = c - 1$. $\square$

Proof of Proposition 44. There are two cases to consider. If the machine contains a real critical job, the proposition follows from Lemma 45. If it contains critical placeholder jobs, the proposition follows from Lemma 46. Finally, if it does not receive placeholder jobs, Lemma 39 bounds its load by $\frac{\delta^2}{1-\delta}B + p_{\text{max}}^{\delta^2 n} \le \left(1 + \frac{\delta^2}{1-\delta}\right)B$. $\square$

We now come to the main result of this section.

Proposition 42. *The makespan of our algorithm is at most $(c + O(\delta))\max(B, L, p_{\text{max}})$ on stable sequences till it employs the Least-Loaded-Strategy (or till the end of the sequence).*

Proof. By Proposition 44 the statement of the lemma holds after the Preparation for the Critical-Job-Strategy. We have to show that it still holds afterwards. There are three cases to consider.

First, consider reserve machines. By Lemma 33 their load is at most $\max((1 + \delta)cB, p_{\text{max}})$ till the Least-Loaded-Strategy is employed.

Second, consider the case that a small job J is scheduled. The job J will be scheduled on a principal machine M with smallest anticipated load. By Lemma 7 said smallest anticipated load is at most $\frac{1}{1-\delta}\tilde{L}$. Since J has size at most $p_{\text{small}}B$, the anticipated load of M won't exceed $\frac{1}{1-\delta}\tilde{L} + p_{\text{small}}B \le (c + \frac{\delta}{1-\delta})\max(B, L)$ after J is scheduled. The last inequality makes use of the fact that $\tilde{L} = L$ for stable sequences, Lemma 38, and that $p_{\text{small}} = c - 1$.

Finally, we consider critical jobs that are scheduled onto principal machines. They replace placeholder-jobs. Such a critical job can have at most $(1 + \delta)$-times the size of the placeholder-job it replaces. Thus it may cause the load of a machine to increase by at most δB. Since each principal machine receives at most two critical jobs the increase on principal machines due to critical jobs is at most $2\delta B$ and the lemma follows.[2] $\square$

Corollary 43. *Till the Least-Loaded-Strategy is used the makespan of our algorithm is at most $(c + O(\delta))\text{OPT}$ on stable sequences.*

Proof. Use Proposition 42 and the fact that the conditions for stable sequences imply that $B = \max\left(p_{\text{max}}^{\delta^2 n}, \hat{L}\right) \le (1+\delta)\text{OPT}$. $\square$

---

[2] A more careful analysis shows that the total increase is in fact most $c\delta B$.

Concerning the case that the algorithm fails. We need to assert certain structural properties if the algorithm FAILS, i.e. reaches line 7 in Algorithm 5. This is done in this section. The first important finding shows that we do not have to deal with huge jobs anymore.

Proposition 47. *If the algorithm* FAILS*, every huge job has been scheduled.*

The second proposition will help us obtain a lower bound on the optimum makespan.

Proposition 48. *If the algorithm* FAILS *at time t we have $\sum_{p\in\mathcal{P}} \tilde{n}_{p,t} w(p) > m$.*

For any job class $p \in \mathcal{P}$ let c'_p denote the number of p-jobs scheduled after the Preparation for the Critical-Job-Strategy, including placeholder-jobs. This is consistent with our notation from Section 5 if we consider the values of c'_p after the execution of Algorithm 4. We call a job class $p \in \mathcal{P}$ *unsaturated* if $c'_p \leq c_p$. Given $p \in \mathcal{P}$, let $\tilde{n}_{p,t}$ denote the number of p-jobs scheduled at any time t including placeholder-jobs. After the sampling phase $\tilde{n}_{p,t} = \max(c'_p, n_{p,t})$.

The most important technical ingredient in this chapter is to establish that if the algorithm FAILS, there is one job class of which a sizable fraction of jobs has not been scheduled even if we take placeholder jobs into account. The next four lemmas prove this by looking at unsaturated job classes.

Lemma 49. *If the algorithm* FAILS *on a stable sequence, there exists an unsaturated job class $p \in \mathcal{P}$. In particular, during the Critical-Job-Strategy every principal machine contains either one big or two medium jobs.*

Proof. Let us assume that every job class is saturated. This implies that at least $n_p - c_p$ jobs of every job class $p \in \mathcal{P}$ fit onto the principal machines. By the properties of stable sequences, at most $n_p - c_p \leq 2m^{3/4}$ jobs of each class thus need to be scheduled onto the reserve machines; that is at most $|\mathcal{P}| \cdot 2m^{3/4}$ in total. By the last condition of stable sequences this is less than $\lfloor \delta m \rfloor$, the number of reserve machines. A contradiction. The algorithm could not have failed.

If there was an unsaturated job class after the Preparation for the Critical-Job-Strategy, $\mathcal{J}_{\mathrm{rep}}$ must have contained precisely m_{empty} elements after the second while-loop in Algorithm 4. Else, another iteration of this loop would have added further elements. Thus, every principal machine that did not already contain real critical jobs received (critical) placeholder-jobs. By Lemma 40 every principal machine in fact received either one big or two medium jobs. $\qquad\square$

Lemma 50. *For every unsaturated job class $p \in \mathcal{P}$, there holds $c'_p \geq \left\lfloor \frac{(1-\delta-2\delta^2)m}{|\mathcal{P}|} \right\rfloor$.*

Proof. Note that $\mathcal{J}_{\mathrm{rep}}$ actually attains cardinality m_{empty} in Algorithm 4, otherwise there could not have been an unsaturated job class. Every time we add an element to $\mathcal{J}_{\mathrm{rep}}$ in line 4 the value c'_p increases for an unsaturated job class p that currently has minimum value c'_p. In particular, whenever we add $|\mathcal{P}|$-many elements to $\mathcal{J}_{\mathrm{rep}}$ the value $\min_{p\in\mathcal{P}\,\mathrm{unsaturated}} c'_p$ increases by at least 1. In total it increases at least $\left\lfloor \frac{m_{\mathrm{empty}}}{|\mathcal{P}|} \right\rfloor$ times. The lemma follows since $m_{\mathrm{empty}} \leq \left(1 - \delta - 2\delta^2\right) m$, see Lemma 41. $\qquad\square$

Lemma 51. *There holds $\sum_{p\in\mathcal{P}} c'_p w(p) \leq m - \lceil \delta m \rceil$.*

Proof. Let n_{med} be the number of medium jobs and n_{big} be the number of big jobs after the Preparation for the Critical-Job-Strategy, then $\sum_{p\in\mathcal{P}} c_p' w(p) = n_{\mathrm{big}} + \frac{n_{\mathrm{med}}}{2}$. But by Lemma 40 every principal machine contains either one big job, two medium jobs or no critical jobs at all after the Preparation for the Critical-Job-Strategy. Reserve machines are empty. Thus, $\sum_{p\in\mathcal{P}} c_p' w(p) \leq |\mathcal{M}| \leq m - \lceil \delta m \rceil$. $\qquad\square$

We now prove one main proposition of this paragraph.

Proposition 48. *If the algorithm* FAILS *at time t we have $\sum_{p\in\mathcal{P}} \tilde{n}_{p,t} w(p) > m$.*

Proof. Let $J = J_t$ be the job that caused the algorithm to fail. Consider the schedule right before job J was scheduled. As a matter of thinking, let us assume that job J resided on some fictional $(m+1)$-th machine $\tilde{M}$ at that time. We award any machine $\frac{1}{2}$ points for each medium job on it and 1 point for each big job on it. This includes placeholder-jobs. Then $\sum_{p\in\mathcal{P}} \tilde{n}_{p,t} w(p)$ is exactly the number of points scored by every machine including $\tilde{M}$.

By Lemma 49 every principal machine scores one point. There was no empty reserve machine, since J could have been scheduled onto it, otherwise. Thus every reserve machine scores at least half a point. We call a machine *bad* if it scored only $1/2$ point. There cannot be two bad reserve machines, since our algorithm would have scheduled any medium job onto such a bad machine rather than creating a second one. Moreover, if there exists a bad machine, job J cannot be medium, i.e. $\tilde{M}$ cannot be not bad, too. We conclude that there is at most one bad machine amongst the $m+1$ machines, which include the fictional machine $\tilde{M}$. All other machines score one point. This implies that $\sum_{p\in\mathcal{P}} \tilde{n}_{p,t} w(p) \geq m + \frac{1}{2}$. $\qquad\square$

Lemma 52. *Assume that the algorithm* FAILS *at time t on a stable sequence and that not all huge jobs are scheduled. Then there exists a job class $p \in \mathcal{P}$ with $\tilde{n}_{p,t} < n_p - 2|\mathcal{P}|m^{3/4}$*

Proof. We first show that there needs to exist a job class $p \in \mathcal{P}$ with $\tilde{n}_{p,t} > c_p' + 2m^{3/4}$. Assume for contradiction sake, that we had $\tilde{n}_{p,t} \leq c_p' + 2m^{3/4}$ for every job class $p \in \mathcal{P}$. Then we get a contradiction to Proposition 48, namely $\sum_{p\in\mathcal{P}} \tilde{n}_{p,t} w(p) \leq \sum_{p\in\mathcal{P}} c_p' w(p) + |\mathcal{P}| \cdot 2m^{3/4} \leq m - \lceil \delta m \rceil + \frac{\delta^3}{2} \left\lfloor \frac{(1-\delta-2\delta^2)m}{|\mathcal{P}|} \right\rfloor \leq m$. The second inequality uses Lemma 51 and the fifth condition on stable sequences.

Thus, let p be such a job class satisfying $\tilde{n}_{p,t} > c_p' + 2m^{3/4}$. Since $\tilde{n}_{p,t} = \max(n_{p,t}, c_p')$, this implies that $n_{p,t} > c_p' + 2m^{3/4}$. Moreover, since $n_{p,t} \leq c_p + 2m^{3/4}$ by the second property of stable sequences, we must have $c_p' < c_p$, i.e. the job class p is unsaturated. Lemma 50 implies $c_p' \geq \left\lfloor \frac{(1-\delta-2\delta^2)m}{|\mathcal{P}|} \right\rfloor$. In particular $n_p \geq c_p > c_p' \geq \left\lfloor \frac{(1-\delta-2\delta^2)m}{|\mathcal{P}|} \right\rfloor$. We conclude that $n_{p,t} \leq n_p - \delta^3 n_p < n_p - \delta^3 \left\lfloor \frac{(1-\delta-2\delta^2)m}{|\mathcal{P}|} \right\rfloor \leq n_p - 2|\mathcal{P}|m^{3/4}$. The first inequality uses the fourth condition of stable sequences, recall that by assumption not all huge jobs are scheduled; the second inequality uses the bound on n_p we just derived; the final inequality uses the fifth condition of stable sequences. $\qquad\square$

We finally prove Proposition 47, the remaining main proposition of this paragraph.

Proposition 47. *If the algorithm* FAILS, *every huge job has been scheduled.*

Proof. Let $\tilde{t}$ be the time the algorithm fails. By Lemma 52 there exists a job class q such that

$$w(q)n_{q,\tilde{t}} < w(q)n_q - w(q)2|\mathcal{P}|m^{3/4} \leq w(q)n_q - |\mathcal{P}|m^{3/4}$$

holds. In particular

$$\sum_{p\in\mathcal{P}} \tilde{n}_{p,\tilde{t}}w(p) \leq \sum_{p\in\mathcal{P}} n_p w(p) - |\mathcal{P}|m^{3/4} \leq \sum_{p\in\mathcal{P}} (n_p - 2m^{3/4})w(p) \leq \sum_{p\in\mathcal{P}} c_p w(p). \tag{1}$$

The first inequality uses the previous bound on $n_{q,\tilde{t}}$ and the fact that for stable sequences $\tilde{n}_{p,\tilde{t}} = \max(c'_p, n_{p,\tilde{t}}) \leq n_p$. For the second inequality observe that $w(p) \leq 1$ for all $p \in \mathcal{P}$. For the last inequality use again the second condition of stable sequences.

Now Proposition 48 and the previous inequality imply that

$$m < \sum_{p\in\mathcal{P}} \tilde{n}_{p,\tilde{t}}w(p) \leq \sum_{p\in\mathcal{P}} c_p w(p).$$

If this was the case, the algorithm would already have chosen the Least-Loaded-Strategy in Algorithm 4 line 6 and thus never failed, i.e. reached line 7 in Algorithm 5. A contradiction. $\square$

The Least-Loaded-Strategy. We now derive two important consequences from the previous section.

Lemma 53. *If the input sequence is stable, the Least-Loaded-Strategy schedules every huge job onto an empty machine. Thus, if the makespan increases due to the Least-Loaded-Strategy scheduling a huge job, it is at most $p_{\max} \leq \text{OPT}$.*

Proof of Lemma 53. By Proposition 47 if a huge job is scheduled using the Least-Loaded-Strategy, our algorithm already decided to do so during the Preparation for the Critical-Job-Strategy, Algorithm 4. At this time all $\lfloor \delta m \rfloor$ reserve machines were empty. By the conditions of stable sequences there are at most $\lfloor \delta m \rfloor$ huge jobs and there will always be an empty reserve machine available once one arrives. $\square$

Lemma 54. *If our algorithm schedules a normal job J using the Least-Loaded-Strategy, the load of the machine the job is scheduled on will be at most $\frac{1}{1-\delta}L + B$. For stable sequences this is at most $\left(2 + \frac{2\delta-\delta^2}{(1-\delta)^2}\right)B = (2 + O(\delta))B$.*

Proof of Lemma 54. Let l be the load of the machine M before job J was scheduled on it. Since M was the least loaded principal machine at that time $l \leq \frac{m}{m-\lfloor\delta m\rfloor}L \leq \frac{1}{1-\delta}L$ by Lemma 4. Since J was normal, its size was at most B. The first part of the lemma follows. For the second part observe that the first condition on stable sequences implies that $L \leq \frac{B}{1-\delta}$ and thus $\frac{1}{1-\delta}L + B \leq \left(2 + \frac{2\delta-\delta^2}{(1-\delta)^2}\right)B = (2 + O(\delta))B$. $\square$

In order to ameliorate this worse general lower bound we need a better upper bound for B.

Lemma 55. *If the Least-Loaded-Strategy is applied on a stable sequence, $B \leq \frac{c}{2}\text{OPT}$.*

Proof. Let us first assert that $\sum_{p\in\mathcal{P}} n_p w(p) \geq \sum_{p\in\mathcal{P}} \tilde{n}_{n,p}^t w(p) > m$. There are two cases. If the algorithm chooses the Least-Loaded-Strategy in the Preparation for the Critical-Job-Strategy there holds $\sum_{p\in\mathcal{P}} c_p w(p) > m$. By the properties of stable sequence $c_p \leq n_p$ and thus the inequality follows. Else, the algorithm FAILS, i.e. reaches line 7 in Algorithm 5. Let t be the time that happens. Then using Proposition 48 we have that $\sum_{p\in\mathcal{P}} n_p w(p) \geq \sum_{p\in\mathcal{P}} \tilde{n}_{n,p}^t w(p) > m$.

Consider any schedule. We say a machine scores $\frac{1}{2}$ points for every medium job and 1 point for every big job. Then $\sum_{p\in\mathcal{P}} n_p w(p) > m$ is the number of points scored in total. Thus there existed a machine which scores strictly more than 1 point. Such a machine must contain either three medium or one big and another critical job. In the former case its load will be at least $3p_{\text{small}}B = 3(c-1)B > \frac{2}{c}B$, in the latter case its load is at least $(p_{\text{small}}B + p_{\text{big}}B) = \left(\frac{3}{2}c - 1\right)B = \frac{2}{c}B$. The last equality holds since $c = \frac{1+\sqrt{13}}{3}$. $\qquad\square$

Final proof of Main Lemma 3 If the algorithm does not change its makespan while applying the Least-Loaded-Strategy, the result follows form Corollary 43. If the makespan of our algorithm is caused by a huge job while applying the Least-Loaded-Strategy, it leads to an optimal makespan of $p_{\max} \leq \text{OPT}$ by Lemma 53. Finally, if the makespan of our algorithm is caused by a normal job it will be $(2 + O(\delta))B$ by Lemma 54. On the other hand, Lemma 55 implies that $\text{OPT} \geq \frac{2}{c}B$ in this case. The competitive ratio is thus at most $\frac{(2+O(\delta))B}{\frac{2}{c}B} \leq c + O(\delta)$.

7 Lower bounds

We establish the following theorem using two lower bound sequences.

Theorem 56. *For every online algorithm A there exists a job set $\mathcal{J}$ such that*

$$\mathbf{P}_{\sigma\sim S_n}\left[A(\mathcal{J}^\sigma) \geq \frac{\sqrt{73}-1}{6}\text{OPT}(\mathcal{J})\right] \geq \frac{1}{6}.$$

This result actually holds for randomized algorithms too if the random choices of the algorithm are included in the previous probability.

Theorem 56 implies the following lower bounds.

Corollary 57. *If an online algorithm A is nearly c-competitive, $c \geq \frac{\sqrt{73}-1}{6} \approx 1.257$.*

Corollary 58. *The best competitive ratio possible in the secretary model is $\frac{\sqrt{73}+29}{36} \approx 1.043$.*

Let us now prove these results. For this section let $c = \frac{\sqrt{73}-1}{6}$ be our main lower bound on the competitive ratio. We consider three types of jobs:

1. *negligible jobs* of size 0 (or a tiny size $\varepsilon > 0$ if one were to insist on positive sizes).
2. *big jobs* of size $1 - \frac{c}{3} = \frac{17-\sqrt{37}}{18} \approx 0.581$.
3. *small jobs* of size $\frac{c}{3} = \frac{1+\sqrt{37}}{18} \approx 0.419$

Let $\mathcal{J}$ be the job set consisting of m jobs of each type.

29

Lemma 59. *There exists a schedule of $\mathcal{J}$ where every machine has load 1. Every schedule that has a machine with smaller load has makespan at least c.*

Proof. This schedule is achieved by scheduling a type 2 and a type 3 job onto each machine. The load of each machine is then 1. Every schedule which allocates these jobs differently must have at least one machine M which contains at least three jobs of type 2 or 3 by the pigeonhole principle. The load of M is then at least $3\frac{c}{3} = c$. □

Given a permutation $\mathcal{J}^\sigma$ of $\mathcal{J}$ and an online algorithm A, which expects $3m + 1$ jobs to arrive in total. Let $A(\mathcal{J}^\sigma, 3m + 1)$ denote its makespan after it processes $\mathcal{J}^\sigma$ expecting yet another job to arrive. Let $P = \mathbf{P}[A(\mathcal{J}^\sigma, 3m + 1) = 1]$ be the probability that A achieves the optimal schedule where every machine has load 1 under these circumstances. Depending on P we pick one out of two input sets on which A performs bad.

Let $j \in \{1, 2\}$. We now consider the job set $\mathcal{J}_j$ consisting of m jobs of each type plus one additional job of type j, i.e. a negligible job if $j = 1$ and a big one if $j = 2$. We call an ordering $\mathcal{J}_j^\sigma$ of $\mathcal{J}_j$ *good* if it ends with a job of type j or, equivalently, if its first $3m$ jobs are a permutation of $\mathcal{J}$. Note that the probability of $\mathcal{J}^\sigma$ being good is $\frac{m+1}{3m+1} \geq \frac{1}{3}$ for $\sigma \sim S_{3m+1}$.

Lemma 60. *We have*

$$\mathbf{P}_{\sigma \sim S_n}\left[A(\mathcal{J}_1^\sigma) \geq c\text{OPT}(\mathcal{J})\right] \geq \frac{1 - P}{3}$$

and

$$\mathbf{P}_{\sigma \sim S_n}\left[A(\mathcal{J}_2^\sigma) \geq c\text{OPT}(\mathcal{J})\right] \geq \frac{P}{3}.$$

Proof. Consider a good permutation of $\mathcal{J}_1$. Then with probability $1 - P$ the algorithm A does have makespan c even before the last job is scheduled. On the other hand $\text{OPT}(\mathcal{J}_1) = 1$. Thus with probability $\frac{1-P}{3}$ we have $A(\mathcal{J}_1^\sigma) = c = c\text{OPT}(\mathcal{J}_1)$.

Now consider a good permutation of $\mathcal{J}_2$. Then, with probability P, algorithm A has to schedule the last job on a machine of size 1. Its makespan is thus $2 - \frac{c}{3} = c^2$ by our choice of c. The optimum algorithm may schedule two big jobs onto one machine, incurring load $2 - \frac{2c}{3} < c$, three small jobs onto another one, incurring load c and one job of each type onto the remaining machines, causing load $1 < c$. Thus $\text{OPT}(\mathcal{J}_2) = c$. In particular we have with probability $\frac{P}{3}$ that $A(\mathcal{J}_2^\sigma) = c^2 = c\text{OPT}(\mathcal{J}_2)$. □

We now conclude the main three lower bound results.

Theorem 56. *For every online algorithm A there exists a job set $\mathcal{J}$ such that*

$$\mathbf{P}_{\sigma \sim S_n}\left[A(\mathcal{J}^\sigma) \geq \frac{\sqrt{73} - 1}{6}\text{OPT}(\mathcal{J})\right] \geq \frac{1}{6}.$$

This result actually holds for randomized algorithms too if the random choices of the algorithm are included in the previous probability.

Proof. By the previous lemma we get that

$$\max_{j=1,2}\left(\mathbf{P}_{\sigma \sim S_n}\left[A(\mathcal{J}_j^\sigma) \geq c\text{OPT}(\mathcal{J})\right]\right) = \max\left(\frac{1 - P}{3}, \frac{P}{3}\right) \geq \frac{1}{6}. \qquad □$$

Corollary 57. *If an online algorithm A is nearly c-competitive, $c \geq \frac{\sqrt{73}-1}{6} \approx 1.257$.*

Proof. This is immediate by the previous theorem. □

Corollary 58. *The best competitive ratio possible in the secretary model is $\frac{\sqrt{73}+29}{36} \approx 1.043$.*

Proof. Let A be any online algorithm. Pick a job set $\mathcal{J}$ according to Theorem 56. Then

$$A^{\mathrm{rom}}(\mathcal{J}) = \mathbf{E}_{\sigma \sim S_n}[A(\mathcal{J}^\sigma)] \geq \frac{1}{6} \cdot \frac{\sqrt{73}-1}{6}\mathrm{OPT}(\mathcal{J}) + \frac{5}{6}\mathrm{OPT}(\mathcal{J}) = \frac{\sqrt{73}+29}{36}\mathrm{OPT}(\mathcal{J}). \quad \square$$

References

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

[2] S. Albers. On randomized online scheduling. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 134–143, 2002.

[3] S. Albers and M. Hellwig. Semi-online scheduling revisited. *Theoretical Computer Science*, 443:1–9, 2012. Publisher: Elsevier.

[4] S. Albers and M. Hellwig. Online makespan minimization with parallel schedules. *Algorithmica*, 78(2):492–520, 2017. Publisher: Springer.

[5] S. Albers and M. Janke. Scheduling in the Random-Order Model. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*. unpublished, 2020.

[6] S. Albers and L. Ladewig. New results for the k-secretary problem. *arXiv preprint arXiv:2012.00488*, 2020.

[7] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg. Matroid Secretary Problems. *Journal of the ACM (JACM)*, 65(6):1–26, 2018. Publisher: ACM New York, NY, USA.

[8] M. Babaioff, N. Immorlica, D. Kempe, and Robert Kleinberg. A knapsack secretary problem with applications. In *Approximation, randomization, and combinatorial optimization. Algorithms and techniques*, pages 16–28. Springer, 2007.

[9] Y. Bartal, A. Fiat, H. Karloff, and R. Vohra. New algorithms for an ancient scheduling problem. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 51–58, 1992.

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

[11] B. Chen, A. van Vliet, and G. J. Woeginger. A lower bound for randomized on-line scheduling algorithms. *Information Processing Letters*, 51(5):219–222, 1994. Publisher: Elsevier.

[12] 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.

[13] 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.

[14] J. Correa, A. Cristi, B. Epstein, and J. Soto. The two-sided game of googol and sample-based prophet inequalities. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2066–2081. SIAM, 2020.

[15] J. Correa, A. Cristi, L. Feuilloley, T. Oosterwijk, and A. Tsigonias-Dimitriadis. The secretary problem with independent sampling. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2047–2058. SIAM, 2021.

[16] J. Correa, P. Dütting, F. Fischer, and K. Schewior. Prophet inequalities for iid random variables from an unknown distribution. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 3–17, 2019.

[17] P. Diaconis and S. Zabell. Closed form summation for classical distributions: variations on a theme of de moivre. *Statistical Science*, pages 284–302, 1991.

[18] J. Dohrau. Online makespan scheduling with sublinear advice. In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 177–188. Springer, 2015.

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

[20] M. Englert, D. Özmen, and M. Westermann. The power of reordering for online minimum makespan scheduling. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 603–612. IEEE, 2008.

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

[22] M. Feldman, O. Svensson, and R. Zenklusen. A simple O (log log (rank))-competitive algorithm for the matroid secretary problem. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 1189–1201. SIAM, 2014.

[23] T. S. Ferguson. Who solved the secretary problem? *Statistical science*, 4(3):282–289, 1989. Publisher: Institute of Mathematical Statistics.

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

[25] G. Galambos and G. J. 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.

[26] O. Göbel, T. Kesselheim, and A. Tönnis. Online appointment scheduling in the random order model. In *Algorithms-ESA 2015*, pages 680–692. Springer, 2015.

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

[28] 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.

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

[30] A. Gupta, R. Mehta, and M. Molinaro. Maximizing Profit with Convex Costs in the Random-order Model. *arXiv preprint arXiv:1804.08172*, 2018.

[31] D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *Journal of the ACM (JACM)*, 34(1):144–162, 1987. Publisher: ACM New York, NY, USA.

[32] H. Kaplan, D. Naori, and D. Raz. Competitive Analysis with a Sample and the Secretary Problem. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2082–2095. SIAM, 2020.

[33] 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.

[34] 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.

[35] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 352–358, 1990.

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

[37] H. Kellerer, V. Kotov, and M. Gabay. An efficient algorithm for semi-online multiprocessor scheduling with given total processing time. *Journal of Scheduling*, 18(6):623–630, 2015.

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

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

[40] 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.

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

[42] N. Korula, V. Mirrokni, and M. Zadimoghaddam. Online submodular welfare maximization: Greedy beats 1/2 in random order. *SIAM Journal on Computing*, 47(3):1056–1086, 2018. Publisher: SIAM.

[43] 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.

[44] D. V. Lindley. Dynamic programming and decision theory. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 10(1):39–51, 1961. Publisher: Wiley Online Library.

[45] 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.

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

[47] V.S. Mirrokni, S. O. Gharan, and M. Zadimoghaddam. Simultaneous approximations for adversarial and stochastic online budgeted allocation. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1690–1701. SIAM, 2012.

[48] 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.

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

[50] J. F. Rudin III. Improved bounds for the on-line scheduling problem. 2001.

[51] J. Sgall. A lower bound for randomized on-line multiprocessor scheduling. *Information Processing Letters*, 63(1):51–55, 1997. Publisher: Citeseer.

A   Missing proofs in Section 4

Lemma 7. *Let $\varphi > 0$ and $t \leq \varphi n$. Then the machine with the k-th least pseudo-load at time t had pseudo-load at most $\frac{m}{m-k+1}\tilde{L}$.*

Proof. Let $\tilde{l}$ be k-th least pseudo-load at time t. This means that there are at least $m - k + 1$ machines with pseudo-load $\tilde{l}_M^t \geq \tilde{l}$. In particular $(m - k + 1)\tilde{l} \leq \sum_M \tilde{l}_M^t \leq m\tilde{L}$. $\qquad\square$

Proposition 8. *If job $J_{\sigma(t+1)}$ is scheduled on the machine M with i-th smallest pseudo-load $\tilde{l}_M^t$ at time t, then, afterwards, its load l_M^{t+1} does not exceed $\left(1 + \left(\frac{m}{m+1-i}\right)\tilde{R}(\mathcal{J})\right)OPT(\mathcal{J})$.*

Proof. Let $\tilde{l}$ be the pseudo-load of the i-th loaded machine before J is scheduled. We have $\tilde{l} \leq \frac{m}{m-i+1}\tilde{L}$ by Lemma 7. Since J had size at most $p_{\max}$, the load of the machine it was scheduled on will not exceed $\tilde{l}+p_{\max} \leq \frac{m}{m-i+1}\tilde{L}+p_{\max} \leq \frac{m}{m-i+1}\frac{\tilde{L}}{\max(L,p_{\max})}\text{OPT}+\text{OPT} = \left(\left(\frac{m}{m-i+1}\right)\tilde{R}(\mathcal{J})+1\right)OPT.$ $\quad\square$

Sampling and the Load Lemma:

Proposition 9. *Let $\mathcal{C} \subset \mathcal{J}$ be a job class for a job set $\mathcal{J}$ of cardinality at least m. Given $\varphi > 0$ and $E \geq 0$ we have*

$$\mathbf{P}_{\sigma \sim S_n}\left[\left|\varphi^{-1}n_{\mathcal{C},\varphi}[\sigma] - n_{\mathcal{C}}\right| \geq E\right] \leq \frac{n_{\mathcal{C}}}{\varphi(E - 1/m)^2}.$$

Proof. For $\sigma \sim S_n$ chosen uniformly randomly, the random variable $n_{\mathcal{C},\varphi}[\sigma]$ is hypergeometrically distributed: It counts how many out of $\lfloor\varphi n\rfloor$ jobs, chosen randomly from the set of all n jobs without replacement, belong to $\mathcal{C}$. The mean of $n_{\mathcal{C},\varphi}$ is thus

$$\mathbf{E}[n_{\mathcal{C},\varphi}] = \frac{\lfloor\varphi n\rfloor}{n}n_{\mathcal{C}},$$

in particular, using that $\mathcal{J}$ has size at least $n \geq m$, we have

$$0 \leq \varphi n_{\mathcal{C}} - \mathbf{E}[n_{\mathcal{C},\varphi}] \leq \frac{1}{n} \leq \frac{1}{m}. \tag{2}$$

Similarly, the variance of $n_{\mathcal{C},\varphi}$ is at most

$$\text{Var}[n_{\mathcal{C},\varphi}] = \frac{n_{\mathcal{C}}(n - n_{\mathcal{C}})\lfloor\varphi n\rfloor(n - \lfloor\varphi n\rfloor)}{n^2(n - 1)} \leq \varphi n_{\mathcal{C}}.$$

By Chebyshev's inequality we have:

$$\mathbf{P}_{\sigma \sim S_n}\left[\left|\varphi^{-1}n_{\mathcal{C},\varphi}[\sigma] - n_{\mathcal{C}}[\mathcal{J}]\right| \geq E\right]$$
$$\leq \mathbf{P}_{\sigma \sim S_n}\left[\left|n_{\mathcal{C},\varphi}[\sigma] - \mathbf{E}[n_{\mathcal{C},\varphi}]\right| \geq \varphi(E - 1/m)\right]$$
$$\leq \frac{\text{Var}[n_{\mathcal{C},\varphi}]}{\varphi^2(E - 1/m)^2}$$
$$\leq \frac{n_{\mathcal{C}}}{\varphi(E - 1/m)^2} \qquad\qquad\square$$

34

Lemma 10. *[Load Lemma [5]] Let $R_{\text{low}} = R_{\text{low}}(m) > 0$, $1 \geq \varphi = \varphi(m) > 0$ and $\varepsilon = \varepsilon(m) > 0$ be three functions such that $\varepsilon^{-4}\varphi^{-1}R_{\text{low}}^{-1} = o(m)$. Then there exists a variable $m(R_{\text{low}}, \varphi, \varepsilon)$ such that we have for all $m \geq m(R_{\text{low}}, \varphi, \varepsilon)$ and all job sets $\mathcal{J}$ with $R(\mathcal{J}) \geq R_{\text{low}}$ and $|\mathcal{J}| \geq m$:*

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

Proof. Let $\delta = \frac{\varepsilon}{2}$ and let $F(m) = \frac{\sqrt{mR_{\text{low}}}\delta}{1+\delta}$. The assumption that $\varepsilon^{-4}\varphi^{-1}R_{\text{low}}^{-1} = o(m)$ already implies $F \in \Theta\left(\sqrt{mR_{\text{low}}}\varepsilon\right) \subset \omega\left(\frac{1}{\varepsilon\sqrt{\varphi}}\right)$.

Let us fix any input sequence $\mathcal{J}$. Given a non-negative integer $j \in \mathbb{Z}_{\geq 0}$ let $p_j = (1+\delta)^{-j}p_{\max}$. For $j > 0$ let $\mathcal{C}_j$ denote the set of jobs in $\mathcal{J}$ that have size in the half-open interval $(p_j, p_{j-1}]$. Note that every job belongs to precisely one job class $\mathcal{C}_j$. Using the notation from Section 4.1 we set $n_j^\sigma = n_{\mathcal{C}_j, \varphi}[\sigma]$ and $n_j = n_{\mathcal{C}_j}$.

Now $L^\downarrow = \frac{1}{m}\sum_{j=1}^\infty (1+\delta)^{-j}n_j p_{\max}$ is the average load if we round down the size of every job in job class $\mathcal{C}_j$ to p_j for every $j \geq 1$. In particular, there holds $L^\downarrow \leq L \leq (1+\delta)L^\downarrow$. Similarly, let $L_\varphi^\downarrow[\sigma] = \frac{\varphi^{-1}}{m}\sum_{j=1}^\infty (1+\delta)^{-j}n_j^\sigma p_{\max}$ be the rounded-down version of L_φ. Again, there holds $L_\varphi^\downarrow[\sigma] \leq L_\varphi[\sigma] \leq (1+\delta)L_\varphi^\downarrow[\sigma]$. Using these approximations, we see that

$$|L_\varphi[\sigma] - L| \leq \left|L_\varphi^\downarrow[\sigma] - L^\downarrow\right| + \max\left(\delta L_\varphi^\downarrow[\sigma], \delta L^\downarrow\right)$$

$$\leq \left|L_\varphi^\downarrow[\sigma] - L^\downarrow\right| + \delta\left|L_\varphi^\downarrow[\sigma] - L^\downarrow\right| + \delta L^\downarrow$$

$$\leq (1+\delta)\left|L_\varphi^\downarrow[\sigma] - L^\downarrow\right| + \delta L$$

We can bound the term in the statement of the lemma via

$$\left|\frac{L_\varphi}{L} - 1\right| \leq (1+\delta)\frac{\left|L_\varphi^\downarrow[\sigma] - L^\downarrow\right|}{L} + \delta. \tag{3}$$

Now, consider the term $\left|L_\varphi^\downarrow[\sigma] - L^\downarrow\right|$. Proposition 9 with $E = (1+\delta)^{j/2}F(m)\sqrt{n_j}$, yields

$$\mathbf{P}_{\sigma \sim S_n}\left[\left|\varphi^{-1}n_j^\sigma - n_j\right| \geq (1+\delta)^{j/2}F(m)\sqrt{n_j}\right] = O\left(\frac{(1+\delta)^{-j}}{\varphi F(m)^2}\right)$$

Consider the event that we have $\left|\varphi^{-1}n_j^\sigma - n_j\right| \geq (1+\delta)^{j/2}F(m)\sqrt{n_j}$ for all j. By the union bound its probability is

$$P(m) = 1 - O\left(\sum_j \frac{(1+\delta)^{-j}}{\varphi F(m)^2}\right) = 1 - O\left(\frac{1}{\delta\varphi F(m)^2}\right) = 1 - o(\varepsilon).$$

The first equality uses the union bound, the second the geometric sequence and the final one the fact, argued at the beginning of the proof, that $F \in \omega\left(\frac{1}{\varepsilon\sqrt{\varphi}}\right)$ and that $\delta = \Theta(\varepsilon)$. Now, if we have

$\left|\varphi^{-1}n_j^\sigma - n_j\right| \geq (1+\delta)^{j/2}F(m)\sqrt{n_j}$ for all $j \geq 1$, we get:

$$\left|L_\varphi^\downarrow[\sigma] - L^\downarrow\right| = \left|\frac{\varphi^{-1}}{m}\sum_{j=0}^\infty (1+\delta)^{-j}n_j^\sigma p_{\max} - \frac{1}{m}\sum_{j=0}^\infty (1+\delta)^{-j}n_j p_{\max}\right|$$

$$\leq \frac{1}{m}\sum_{j=0}^\infty (1+\delta)^{-j}\left|\varphi^{-1}n_j^\sigma - n_j\right|p_{\max}$$

$$\leq \frac{\sqrt{p_{\max}}}{m}\sum_{j=0}^\infty (1+\delta)^{-j/2}F(m)\sqrt{n_j}\sqrt{p_{\max}}$$

$$\leq \frac{\sqrt{p_{\max}}F(m)}{\sqrt{m}}\left(\frac{1}{m}\sum_{j=0}^\infty (1+\delta)^{-j}n_j p_{\max}\right)^{1/2}$$

$$\leq \frac{\sqrt{L}F(m)}{\sqrt{R_{\mathrm{low}}m}}\sqrt{L^\downarrow}$$

$$\leq \frac{F(m)}{\sqrt{R_{\mathrm{low}}m}}L = \frac{\delta}{1+\delta}L.$$

The first inequality is the triangle inequality, the second holds per assumption, the third is the Cauchy–Schwarz inequality, the fourth uses the definition of $L^\downarrow$ and the fact that $\sqrt{p_{\max}} = \sqrt{L/R[\mathcal{J}]} \leq \sqrt{L/R_{\mathrm{low}}}$, the last inequality uses that $L^\downarrow \leq L$ and the final equality is simply the definition of F.

Combining this with Equation (3) yields that we have with probability $P(m) \in 1 - o(\varepsilon)$

$$\left|\frac{L_\varphi}{L} - 1\right| \leq (1+\delta)\frac{\left|L_\varphi^\downarrow[\sigma] - L^\downarrow\right|}{L} + \delta \leq 2\delta = \varepsilon.$$

It thus suffices to choose $m(R_{\mathrm{low}}, \varphi, \varepsilon)$ such that $P(m) \leq 1 - \varepsilon$ for all $m \geq m(R_{\mathrm{low}}, \varphi, \varepsilon)$. $\qquad\square$

B   Missing proofs in Section 4.2

Corollary 12. *Let $\mathcal{J}^\sigma$ be any (ordered) input sequence and let $L_{\mathrm{guess}} \geq L[\mathcal{J}]$. Then the makespan of* LightLoad$[L_{\mathrm{guess}}]$ *is at most* $1.75 \cdot \max(L_{\mathrm{guess}}, \mathrm{OPT}(\mathcal{J}))$.

Proof of Corollary 12. Let wlog. $\mathcal{J}^\sigma = J_1, \ldots, J_n$ and set $\mathrm{OPT} = \mathrm{OPT}(\mathcal{J}^\sigma) = \mathrm{OPT}(J_1, \ldots, J_n)$. We append a certain number of jobs $J_{n+1}, \ldots, J_{n'}$ to the sequence such that the average load of $J_1, \ldots, J'_n$ is L_{guess} and $\mathrm{OPT}(J_1, \ldots, J'_n) = \max(L_{\mathrm{guess}}, \mathrm{OPT})$. We use the following procedure to construct the sequence:

Algorithm 6 Appending a certain job sequence.

1: Start with $n' = n$ and any optimal schedule of $J_1, \ldots, J_n$.

2: **while** $L_{n'} = \frac{1}{m}\sum_{i=1}^{n'} p_{n'} < L_{\mathrm{guess}}$ **do**

3:     Let M be a least loaded machine and l be its load.

4:     Append job $J_{n'+1}$ of size $p'_n = \min(L_{\mathrm{guess}} - l, m(L_{\mathrm{guess}} - L_{n'}))$ to the sequence.

5:     Schedule $J_{n'}$ onto M. $n' \leftarrow n' + 1$.

It is easy to see that the previous schedule outputs a job sequence of average load $L[J_1, \ldots, J_{n'}] = L_{n'} = L_{\text{guess}}$, assuming it started with a sequence $J_1, \ldots, J_n$ of average load at most L_{guess}. Furthermore it maintains a schedule with makespan at most $\max(L_{\text{guess}}, \text{OPT})$. This is necessarily an optimal schedule, since both the average load L_{guess} as well as the optimum $\text{OPT} = \text{OPT}(J_1, \ldots, J_n)$ of a prefix are lower bounds on the optimum makespan. We've thus shown that $\text{OPT}(J_1, \ldots, J_{n'}) = \max(L_{\text{guess}}, \text{OPT})$.

Since we can apply Theorem 11 to $J_1, \ldots, J_{n'}$ we get:

$$\text{LightLoad}[L_{\text{guess}}](J_1, \ldots, J_n) \leq \text{LightLoad}[L[J_1, \ldots, J_{n'}]](J_1, \ldots, J_{n'})$$
$$\leq 1.75\text{OPT}(J_1, \ldots, J_{n'})$$
$$= 1.75 \max(L_{\text{guess}}, \text{OPT}). \qquad \square$$

C   Second reduction. Full proof of Main Lemma 2

Throughout this proof we assume that all job sets $\mathcal{J}$ considered are proper. Many notations in this proof will depend on the job set $\mathcal{J}$, the number of machines m and possibly the job order. For simplicity we omit these dependencies whenever possible. If needed, we include it using the notation $\mathcal{P}[\mathcal{J}^\sigma]$ or even $\mathcal{P}[\mathcal{J}^\sigma, m]$, $\hat{n}_p[\mathcal{J}^\sigma]$, etc. In particular, we write mostly δ for the function $\delta(m) = \frac{1}{\log(m)}$. It is very important to note for the arguments in this section, that this is a function in m whose inverse grows sub-polynomially in m.

For every job set $\mathcal{J}$ we fix a set $S = S[\mathcal{J}, m] \subset \mathcal{J}$ consisting of the $\lceil \delta(m)^{-7/3} \rceil$ largest jobs. We solve ties arbitrarily. Technically, we could choose any exponent other than $7/3$ in the open interval $(2, 3)$, too. Let $s_{\min} = s_{\min}[\mathcal{J}, m]$ be the size of the smallest job in the set S. Recall the geometric rounding function $f(p) = (1 + \delta)^{\lfloor \log_{1+\delta} p \rfloor}$ and consider the set $\mathcal{P}_{\text{glob}} = \mathcal{P}_{\text{glob}}[\mathcal{J}, m] = \{f(p_t) \mid p_t \text{ is the size of any job } J_t \in \mathcal{J}\}$ of all rounded sizes of jobs in $\mathcal{J}$. Then consider the subset $\hat{\mathcal{P}} = \hat{\mathcal{P}}[\mathcal{J}, m] = \{(1 + \delta)^i \in \mathcal{P}_{\text{glob}} \mid p_{\text{small}} \max(s_{\min}, (1 - \delta)L) < (1 + \delta)^i\}$. We will see that this set is likely a superset of $\mathcal{P}[\mathcal{J}^\sigma, m]$, which does not depend on the job order.

The following estimates of the sizes of $\mathcal{P}$ and $\hat{\mathcal{P}}$ will be relevant later.

Lemma 61. *We have* $|\mathcal{P}| \leq 1 - \lfloor \log_{1+\delta}(p_{\text{small}}) \rfloor \leq O\left(\delta^{-1}\right)$.

Proof. First observe that $\mathcal{P}$ contains precisely one element for each power of $(1 + \delta)$ in the half-open interval $((1 + \delta)^{-1}p_{\text{small}}B, B]$. In particular $|\mathcal{P}| \leq 1 - \lfloor \log_{1+\delta}(p_{\text{small}}) \rfloor \leq O\left(\delta^{-1}\right)$. $\qquad \square$

Lemma 62. *We have* $|\hat{\mathcal{P}}| \leq \delta(m)^{-7/3} - \lfloor \log_{1+\delta}(p_{\text{small}}) \rfloor \leq O(\delta^{-7/3})$

Proof. Indeed, there are precisely $1 - \lfloor \log_{1+\delta}(p_{\text{small}}) \rfloor$ powers of $(1 + \delta)$ in the half-open interval $(p_{\text{small}}s_{\min}, s_{\min}]$, in particular $\hat{\mathcal{P}}$ contains at most that many elements of size lesser or equal to $s_{\min}$. Now all elements in $\hat{\mathcal{P}}$ which have size strictly greater than $s_{\min}$ need to be the rounded sizes of jobs in S excluding the smallest job in S. Thus, there are at most $\delta(m)^{-7/3} - 1$ elements in $\hat{\mathcal{P}}$ of size strictly greater than $s_{\min}$. In particular $|\hat{\mathcal{P}}| \leq 1 - \lfloor \log_{1+\delta}(p_{\text{small}}) \rfloor + \delta(m)^{-7/3} - 1$. $\qquad \square$

For every $p \in \hat{\mathcal{P}}[\mathcal{J}, m]$ we consider the job class $\mathcal{C}_p = \mathcal{C}_p[\mathcal{J}, m] \subset \mathcal{J}$ of jobs whose rounded size is p. Using the notation from Section 4.1 we set $n_{p,\varphi} = n_{\mathcal{C}_p,\varphi}$ and $n_p = n_{\mathcal{C}_p}$ for every $p \in \hat{\mathcal{P}}$ and $0 < \varphi < 1$. We defined the property of being stable in a way that lends itself to algorithmic applications. We now give similar, in fact slightly stronger, conditions better suited for a probabilistic arguments.

37

Definition 63. *We call a proper job sequence $\mathcal{J}^\sigma$ probabilistically stable if the following holds:*

1. *The load estimate $\hat{L} = \hat{L}_{\delta^2}[\mathcal{J}^\sigma]$ for $L = L[\mathcal{J}]$ is good, i.e. $(1-\delta)L \leq \hat{L} \leq (1+\delta)L$.*
2. *There is at least one job in S among the $\lceil \delta^2 n \rceil$ first jobs in $\mathcal{J}^\sigma$.*
3. *For every $p \in \hat{\mathcal{P}}$ we have $\left| \delta^{-2} n_{p,\delta^2} - n_p \right| \leq m^{3/4} - 1$.*
4. *For every $p \in \mathcal{P}_{\mathrm{med}}[\mathcal{J}^\sigma]$ we have $2n_{p,\delta^2} \leq n_p$.*
5. *Let $t_S = t_S[\mathcal{J}^\sigma, m]$ be the time the last job in S arrived, then $t_S \leq \left(1 - \delta(m)^{8/3}\right) n$.*
6. *For every $p \in \hat{\mathcal{P}}$ with $n_p > \left\lfloor \frac{(1-\delta-2\delta^2)m}{|\mathcal{P}|} \right\rfloor$ there holds $n_{p,1-\delta^{8/3}} \leq \left(1 - \delta^3\right) n_p$.*

We refer to these six conditions as probabilistic conditions.

The following lemma is shows that we can analyze the probability of a sequence being probabilistically stable instead of using the conditions from Section 6.2.

Lemma 64. *There exists a number m_0 such that for all $m \geq m_0$ every probabilistically stable sequence is stable.*

The constant m_0 in the previous lemma comes from the following technical lemma.

Lemma 65. *There exists $m_0 > 0$, such that for all $m \geq m_0$ and all proper job sets $\mathcal{J}$ we have*
$$\delta(m)^{-7/3} \leq \lfloor \delta m \rfloor \ \text{ and } \ \delta(m)^3 \cdot \left\lfloor \frac{(1-\delta(m)-2\delta(m)^2)m}{|\mathcal{P}(m,\mathcal{J})|} \right\rfloor \geq 2|\mathcal{P}(m,\mathcal{J})|m^{3/4}.$$

Proof. This comes down to asymptotic observations. For the first inequality use that $\delta(m)^{-7/3} = \log^{7/3}(m) = o(m)$ while $\lfloor \delta m \rfloor = \Theta(m)$.

For the second inequality observe that $|\mathcal{P}(m,\mathcal{J})| = O(\delta(m)^{-1}) = O(\log(m))$ by Lemma 61. Then we can see that $\delta(m)^3 \cdot \left\lfloor \frac{(1-\delta(m)-2\delta(m)^2)m}{|\mathcal{P}(m,\mathcal{J})|} \right\rfloor = \Omega(\delta(m)^4 m) = \Omega\left(\frac{m}{\log^4(m)}\right)$ while, on the other hand, $2|\mathcal{P}|m^{3/4} = O\left(\log(m)m^{3/4}\right) = o\left(\frac{m}{\log^4(m)}\right)$.

These asymptotic observations already imply the statement of the lemma. $\square$

Proof of Lemma 64. We consider the five conditions of stable sequences separately
1. The first condition of stable sequences agrees with the first probabilistic condition.
2. First consider job classes $p \in \mathcal{P}[\mathcal{J}^\sigma, m] \setminus \mathcal{P}_{\mathrm{glob}}[\mathcal{J}, m]$. For these job classes there holds $n_p = 0$. This already implies that $c_p = 0$ holds, too. Thus, the second condition follows trivially for these job classes.

By the second probabilistic condition, we have that $p_{\max}^{\varphi n}[\mathcal{J}^\sigma] \geq s_{\min}$ and by the first probabilistic condition there holds $\hat{L}[\mathcal{J}^\sigma] \geq (1-\delta)L$. In particular $B[\mathcal{J}^\sigma] = \max\left(p_{\max}^{\varphi n}, \hat{L}\right) \geq \max(s_{\min}, (1-\delta)L)$ and thus $\mathcal{P}(\mathcal{J}^\sigma, m) \cap \mathcal{P}_{\mathrm{glob}} \subseteq \hat{\mathcal{P}}(\mathcal{J}, m)$. There are two cases to consider now. If $c_p = \left\lfloor \left(\delta^{-2}\hat{n}_p - m^{3/4}\right) w(p) \right\rfloor w(p)^{-1}$ we conclude, using the third probabilistic condition, that $\left|(c_p + m^{3/4}) - n_p\right| \leq |\delta^{-2}\hat{n}_p - n_p + 1| \leq m^{3/4}$, which already implies that $c_p \leq n_p \leq c_p + 2m^{3/4}$. If $c_p = \hat{n}_p w(p)^{-1} \geq \left\lfloor \left(\delta^{-2}\hat{n}_p - m^{3/4}\right) w(p) \right\rfloor$ the second bound $n_p \leq c_p + 2m^{3/4}$ still holds. The first bound, $c_p \hat{n}_p w(p)^{-1} \leq n_p$ is trivial if $w(p)^{-1} = 1$, or, equivalently, if $p \notin \mathcal{P}_{\mathrm{med}}[\mathcal{J}^\sigma]$. Else, it follows from the fourth probabilistic condition.
3. To conclude the third condition of stable sequences note that the second probabilistic condition implies that all huge jobs have size strictly greater than $s_{\min}$. This implies that they lie in

38

S and that there are at most $|S| = \delta(m)^{-7/3}$ many of those jobs. Since we only consider $m \geq m_0$ we have $\delta(m)^{-7/3} \leq \lfloor \delta m \rfloor$ by Lemma 65. Hence, the third condition of stable sequences follows.

4. Consider $p \in \mathcal{P}(\mathcal{J}^\sigma, m)$ with $n_p > \left\lfloor \frac{(1-\delta-2\delta^2)m}{|\mathcal{P}|} \right\rfloor$. As we argued when proving the second condition $p \in \hat{\mathcal{P}}(\mathcal{J}, m)$. By the second probabilistic condition all huge jobs lie in S since they have size strictly greater than $\hat{B} \geq p_{\max}^{\varphi n} \geq s_{\min}$. It thus suffices to show that at most $(1-\delta^3)n_p$ jobs arrived at time $t_S[\mathcal{J}^\sigma, m]$, the time the last job in S arrived. But by the fifth probabilistic condition this value is at most $n_{p,1-\delta^{8/3}}$, which is less than $(1-\delta^3)n_p$ by the sixth probabilistic condition. The fourth condition for stable sequences follows.

5. Finally, the fifth condition of stable sequences is already a consequence of choosing $m \geq m_0$ and Lemma 65. $\qquad\square$

Now, we analyze the probability of each probabilistic condition separately. Namely, we consider

$$P_i(m) = \sup_{\mathcal{J} \text{ proper}} \mathbf{P}_{\sigma \sim S_n}\left[\mathcal{J}^\sigma \text{does not fulfill the } i\text{-th condition}\right].$$

Recall, that $P(m)$ similarly defines the worst probability with which a sequence may not be stable. It is the value we are interested in. The values $P_i(m)$ relate to $P(m)$ by the following corollary, which is an immediate consequence of Lemma 64 and the union bound.

Corollary 66. *We have that $P(m) \leq \sum_{i=1}^{6} P_i(m)$ for all $m \geq m_0$ if we choose m_0 as in Lemma 65.*

Thus, we are left to see that all the $P_i(m)$ vanish.

Lemma 67. *For every i we have $\lim_{m \to \infty} P_i(m) = 0$.*

Proof. We again consider every choice of $1 \leq i \leq 6$ separately.

1. Apply the Load Lemma, Lemma 10, with $R_{\text{low}} = \frac{(1-\delta)\delta^3}{2(\delta^2+1)}(2-c)$, $\varphi = \delta^2$ and $\varepsilon = \delta$. Then for m large enough, there holds $\mathbf{P}_{\sigma \sim S_n}\left[\left|\frac{L_{\delta^2}[\mathcal{J}^\sigma]}{L[\mathcal{J}]} - 1\right| \geq \delta\right] < \delta$. Note that the condition $\left|\frac{L_{\delta^2}[\mathcal{J}^\sigma]}{L[\mathcal{J}]} - 1\right| \geq \delta$ is equivalent to $(1-\delta)L \leq \hat{L} \leq (1+\delta)L$. Thus $P_1(m) \leq \delta(m)$ and in particular $\lim_{m \to \infty} P_1(m) = 0$.

2. Let $J \in S$. The probability that J is not among the $\lfloor \delta^2 m \rfloor$ first jobs is at most $1-\delta^2$ after random permutation. In particular the probability $P_2(m)$ that none of the jobs in S is among the first $\lfloor \delta^2 m \rfloor$ jobs can be bounded via $(1-\delta^2)^{|S|} \leq \frac{1}{1+\delta^2|S|}$ using Bernoulli's inequality. Thus $P_2(m) \leq \frac{1}{1+\delta(m)^2|S|} \leq \frac{1}{1+\delta(m)^{-1/3}} \leq \delta(m)^{1/3}$ which tends to 0 for $m \to \infty$.

3. Fix $p \in \hat{\mathcal{P}}$. By Proposition 9 we have $\mathbf{P}_{\sigma \sim S_n}\left[\left|\delta^{-2}n_{p,\delta^2} - n_p\right| \geq m^{3/4} - 1\right] \leq \frac{n_p}{\delta^2(m^{3/4}-1-1/m)^2}$ and thus by the union bound $P_3(m) \leq \frac{\sum_{p \in \hat{\mathcal{P}}} n_p}{\delta^2(m^{3/4}-1-1/m)^2}$. Now observe that there holds $L \geq \frac{1}{m}\sum_{p \in \hat{\mathcal{P}}} p \cdot n_p \geq \frac{1}{m}\sum_{p \in \hat{\mathcal{P}}} p_{\text{small}}(1-\delta)L \cdot n_p$, thus $\sum_{p \in \hat{\mathcal{P}}} n_p \leq \frac{m}{(1-\delta)p_{\text{small}}}$. From this we conclude that $P_3(m) \leq \frac{m}{(1-\delta)p_{\text{small}}\delta^2(m^{3/4}-1-1/m)^2} = O\left(\frac{1}{\sqrt{m}}\right)$ which already shows that $\lim_{m \to \infty} P_3(m) = 0$.

4. Recall that f is the geometric rounding function. Let $\hat{B}[\mathcal{J}^\sigma] = \max((1+\delta)L, f(B[\mathcal{J}^\sigma]))$ and let $\hat{\mathcal{P}}_{\text{med}}[\mathcal{J}^\sigma] = \{p \in \mathcal{P}_{\text{glod}} \mid (1+\delta)^{-3}p_{\text{small}}\hat{B}[\mathcal{J}^\sigma] \leq p \leq p_{\text{big}}\hat{B}[\mathcal{J}^\sigma]\}$. We have $\mathcal{P}_{\text{med}}[\mathcal{J}^\sigma] \subset \hat{\mathcal{P}}_{\text{med}}[\mathcal{J}^\sigma]$ if the first probabilistic condition holds, i.e. $(1-\delta)L \leq \hat{L}[\mathcal{J}^\sigma] \leq (1+\delta)L$. Since the probability of the latter not being true is $P_1(m)$ and vanishes for $m \to \infty$ it suffices to consider the second probabilistic condition where we replace $\mathcal{P}_{\text{med}}[\mathcal{J}^\sigma]$ with $\hat{\mathcal{P}}_{\text{med}}[\mathcal{J}^\sigma]$.

Now let $\hat{B}_{\text{fix}}$ be any possible value the variable $\hat{B}[\mathcal{J}^\sigma]$ may obtain. We want to condition ourselves on the case that $\hat{B}[\mathcal{J}^\sigma] = \hat{B}_{\text{fix}}$ for $\hat{B}_{\text{fix}}$ either $(1+\delta)L$ or the rounded size of some element in S exceeding $(1+\delta)L$. Note, that $\hat{B}[\mathcal{J}^\sigma]$ may attain other values than the previously mentioned ones only if either the first or second probabilistic condition is not met. Since $P_1(m) + P_2(m) \to 0$ we may ignore these cases. Fixing $\hat{B}$ also fixes the set $\hat{\mathcal{P}}_{\text{med}}$.

Consider $p \in \hat{\mathcal{P}}_{\text{med}}$. We want an upper bound on $\mathbf{E}[n_{p,\delta^2} \mid \hat{B} = \hat{B}_{\text{fix}}]$, the expected value of n_{p,δ^2} conditioned on our choice of $\hat{B}$. What does it mean to condition on $\hat{B}[\mathcal{J}^\sigma] = B_{\text{fix}}$? It is simply equivalent to stating that no element in S of rounded size strictly greater than B_{fix} occurs in the sampling phase and that either an element of rounded size B_{fix} occurs in the sampling phase or $B_{\text{fix}} = (1+\delta)L$. Thus conditioning on $\hat{B}[\mathcal{J}^\sigma] = B_{\text{fix}}$ just fixes the position of some of the $\lceil \delta(m)^{-7/3} \rceil$ elements in S. The expected value of n_{p,δ^2} is maximized if we consider the case where all elements of S have to occur after the sampling phase. In this case, n_{p,δ^2} is hypergeometrically distributed. We sample $\delta^2 n$ jobs from a set of $n - |S|$ jobs and count the number of p jobs. Thus, we see that $\mathbf{E}[n_{p,\delta^2} \mid \hat{B} = \hat{B}_{\text{fix}}] \leq \frac{\lceil \delta^2 n \rceil n_p}{n - \lceil \delta(m)^{-7/3} \rceil} \leq 2\delta^2 n_p$. For the latter inequality we need to choose m (and thus $n \geq m$) sufficiently large. Now, we get by Markov's inequality

$$\mathbf{P}\left[n_{p,\delta^2} \geq \frac{n_p}{2}\right] \leq \mathbf{P}\left[n_{p,\delta^2} \geq \frac{\mathbf{E}[n_{p,\delta^2}]}{4\delta^2}\right] \leq 4\delta^2.$$

We have $|\hat{\mathcal{P}}_{\text{med}}[\mathcal{J}^\sigma]| \leq \log_{1+\delta}(p_{\text{big}}) - \log_{1+\delta}(p_{\text{small}}) + 4 = O(\delta(m))$. Thus, by the union bound $P_4(m) \leq |\hat{\mathcal{P}}_{\text{med}}| \cdot 4\delta^2 = O((\delta(m))^{-1})$ if we condition ourselves on $\hat{B}[\mathcal{J}^\sigma] = \hat{B}_{\text{fix}}$. Since this holds for all possible choices of $\hat{B}_{\text{fix}}$ but few degenerate ones that occur with probability at most $P_1(m) + P_2(m)$ we get that $P_4(m) \leq P_1(m) + P_2(m) + O(\delta(m))$. Since all these terms vanish for $m \to \infty$ we also have that $\lim_{m \to \infty} P_4(m) = 0$.

5. Let $l = \lceil \delta(m)^{-7/3} \rceil$, then there holds $P_5(m) = 1 - \prod_{i=1}^{l} \frac{\lfloor (1-\delta(m)^{8/3})n \rfloor - i}{n-i}$. Indeed, if we choose any order $S = \{s_1, \ldots, s_l\}$ then the i-th term in the product denotes the probability that the i-th element s_i is among the $\lceil (1 - \delta(m)^{8/3})n \rceil$-last elements conditioned on the earlier elements already fulfilling this condition. But now we see using Bernoulli's inequality that $\prod_{i=1}^{l} \frac{\lfloor (1-\delta(m)^{8/3})n \rfloor - i}{n-i} \geq \left(1 - \delta(m)^{8/3} - \frac{l+1}{n}\right)^l \geq 1 - \delta(m)^{8/3}l - \frac{l(l+1)}{n}$. Since for proper job sets $n \geq m$, this implies $P_5(m) \leq \delta(m)^{8/3}l - \frac{l(l+1)}{m} = O(\delta(m)^{1/3})$. In particular $\lim_{m \to \infty} P_5(m) = 0$.

6. Fix any $p \in \hat{\mathcal{P}}$ with $n_p > \left\lfloor \frac{(1-\delta-2\delta^2)m}{|\mathcal{P}|} \right\rfloor = \Omega(m)$. Then we have by Proposition 9

$$\mathbf{P}\left[n_{p,1-\delta^{8/3}} \geq \left(1-\delta^3\right) n_p\right] \leq \mathbf{P}\left[\left|(1-\delta^{8/3})^{-1}n_{p,1-\delta^{8/3}} - n_p\right| \geq \frac{\delta^{8/3}(1-\delta^{1/3})n_p}{1-\delta^{8/3}}\right]$$

$$\leq \frac{n_p}{(1-\delta^{8/3})\left(\frac{\delta^{8/3}(1-\delta^{1/3})n_p}{1-\delta^{8/3}} - 1/m\right)^2}$$

$$= O\left(\frac{\delta^{-16/3}}{n_p}\right) = O\left(\frac{\delta^{-16/3}}{m}\right).$$

By the union bound and Lemma 62 there holds that $P_6(m) = O(|\hat{\mathcal{P}}|\delta^{-16/3}/m) = O(\delta^{-23/3}/m)$. Thus, $\lim_{m \to \infty} P_6(m) = 0$. $\qquad\square$

Proof of Main Lemma 2. By Corollary 66 we have $P(m) \leq \sum_{i=1}^{6} P_i(m)$ for $m \geq m_0$ and by Lemma 67 we have that $\lim_{m \to \infty} \sum_{i=1}^{6} P_i(m) = 0$. Thus $\lim_{m \to \infty} P(m) = 0$. $\qquad\square$