Scheduling in the Secretary Model

Susanne Albers
Department of Computer Science, Technische Universität München, Germany

Maximilian Janke
Department of Computer Science, Technische Universität München, Germany

Abstract

This paper studies online makespan minimization in the secretary model. Jobs, specified by their processing times, are presented in a uniformly random order. The input size n is known in advance. An online algorithm has to non-preemptively 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 [4] provides a very simple approach 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.

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.

2012 ACM Subject Classification Theory of computation → Online algorithms

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

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2021.6

Related Version *Full Version*: `https://arxiv.org/abs/2103.16340`

Funding Work Supported by the European Research Council, Grant Agreement No. 691672, Project APEG.

1 Introduction

We study one of the most basic scheduling problems, 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 completion time of the last job. This problem admits a long line of research and countless practical applications in both, its offline variant see e.g. [31, 34] 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 $\mathrm{OPT}(\mathcal{J})$ 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 $\frac{A(\mathcal{J}^\sigma)}{\mathrm{OPT}(\mathcal{J})}$. The maximum ratio, $c = \sup_{\mathcal{J},\sigma} \frac{A(\mathcal{J}^\sigma)}{\mathrm{OPT}(\mathcal{J})}$, 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 [21, 24, 25, 35, 44, 46, 47] and references therein.

This paper studies 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* is $c = \sup_{\mathcal{J}} \mathbf{E}_\sigma \left[\frac{A(\mathcal{J}^\sigma)}{\mathrm{OPT}(\mathcal{J})}\right] = \sup_{\mathcal{J}} \frac{1}{n!} \sum_{\sigma \in S_n} \frac{A(\mathcal{J}^\sigma)}{\mathrm{OPT}(\mathcal{J})}$, 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* to set this result apart from [6] where we provide a 1.8478-competitive 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 scheduling algorithm designed with random-order arrival in mind [3, 28, 51, 52]. We hope to raise attention to these two surprisingly different models. Even though for the adversarial model such information is useless; the secretary-model requires novel and significantly different approaches and leads to, as our results show, vastly better performance guarantees.

Frameworks similar to the secretary model 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 [29, 36, 38, 48], various generalizations of the secretary problem [9, 24, 25, 33, 35, 44, 46], knapsack problems [10], bin packing [42], facility location [49], packing LPs [43], convex optimization [32], welfare maximization [45], budgeted allocation [50] and recently scheduling [3, 6, 28, 51, 52]. We refer to the chapter [8] for a general overview over random-order models.

For makespan minimization, the role of randomization is poorly understood. The lower bound of 1.581 from [14, 55] is considered pessimistic and exhibits quite a big gap towards the best randomized ratio of 1.916 from [2]. A main consequence of the paper is that random-order arrival allows to beat the lower bound of 1.581. This formally sets the secretary 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 deterministic adversarial setting. For m identical machines, Graham [31] 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 [27, 11, 37, 1] leading to the currently best competitive ratio by Fleischer and Wahl [26], which approaches 1.9201 for $m \to \infty$. Chen et al. [15] presented a deterministic algorithm whose competitive ratio is at most $(1+\varepsilon)$-times the optimum one, although the actual ratio remains to be determined. For general m, lower bounds are provided in [23, 12, 30, 53]. The currently best bound is due to Rudin III [53] 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 $m \to \infty$ [14, 55]. There has been considerable research interest in tightening these bounds.

Recent results for makespan minimization consider variants where the online algorithm obtains extra resources. In semi-online settings, additional information on the job sequence is given in advance, such as the optimum makespan [13, 39] or the total processing time of jobs [4, 16, 41, 40]. In the former model, the optimum competitive ratio lies in the interval $[1.333, 1.5]$, see [13], while for the latter the optimum competitive ratio is known to be 1.585 cf. [4, 40]. Taking this further, the advice complexity setting allows the algorithm to receive a certain number of advice bits from an offline oracle [5, 20, 41]. Other algorithms can migrate jobs [54] or offer a buffer, which they use to reorder the job sequence [22, 41].

The secretary problem is even older than scheduling [25]. We only summarize the work most relevant to this paper. Lindley [47] and Dynkin [21] 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 [7, 44] or even matroids [9, 24, 46]. A modern version considers adversarial orders but allows prior sampling [18, 35, 8, 33]. Related models are prophet inequalities and the game of googol [17, 19].

So far, little is known for scheduling in the secretary model. Osborn and Torng [52] prove that Graham's greedy strategy is still not better than 2-competitive for $m \to \infty$. We study makespan minimization in the restricted random-order model where n is not known in advance [6] and the dual problem, Machine Covering, in the secretary model [3]. Molinaro [51] studies a very general scheduling problem. His algorithm uses n to restart itself after half the jobs are seen and has expected makespan $(1+\varepsilon)\mathrm{OPT} + O(\log(m)/\varepsilon)$. Göbel et al. [28] study scheduling on a single machine where the goal is to minimize weighted completion times. Their competitive ratio is $O(\log(n))$ whereas they show that adversarial models allow no sublinear competitive ratios.

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 [4] to be 1.75-competitive. A more sophisticated algorithm vastly improves this competitive ratio to 1.535. Both algorithms are deterministic. This ratio of 1.535 beats all lower bounds for adversarial scheduling, including the bound of 1.582 for randomized algorithms. [14, 55]

Our main results focus on large number of machines, $m \to \infty$. This is in line with most recent adversarial results [3, 2, 26] and all random-order scheduling results [6, 28, 51, 52], excluding [28] who study scheduling on one machine. While adversarial 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 small number of machines. We prove that our simple algorithm is $\big(1.75 + O(\frac{1}{\sqrt{m}})\big)$-competitive. Explicit bounds on the term hidden in the O-notation are provided. This result indicates that the focus of contemporary analyses on the limit case is sensible and does not hide unreasonably large terms.

All upper bounds in this paper abide to the stronger measure of *nearly competitiveness* from [6]. An algorithm is required to achieve its competitive ratio not only in expectation but on nearly all input permutations. Thus, input sequences where the competitive ratios are not obtained can be considered extremely rare and pathological. Moreover, we require worst-case guarantees even for such pathological inputs. This is relevant to practical applications, where we do not expect fully random inputs. Both algorithms hold up to this stronger measure of nearly competitiveness.

A basic approach in secretary models uses sampling statistics; a small part of the input allows to predict the rest. Sampling lets us include techniques from semi-online and advice settings 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 [33, 16, 41, 40] to our model. These algorithms need to know the total processing volume right from the start. The advanced algorithm in this paper out-competes the optimum competitive ratio of 1.585 these semi-online algorithms can achieve [1, 40]. We conjecture that this is not possible for order oblivious algorithms that solely use sampling. Order oblivious algorithms first observe a random sample and then treat the input sequence in an adversarial order [8, 33]. Our analysis indicates that LightLoad can be adapted as an order-oblivious algorithm. The 1.535-competitive algorithm does not maintain its competitive ratio in such a setting.

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 follow three steps, which leads to the situation depicted in Figure 1. In the first step, adversarial analyses give worst-case guarantees and take care of *simple job sets*. These simple sets lack structure to be exploited via random reordering but do not pose problems to online algorithms. We thus are reduced to non-simple inputs. Non-simple random sequences have useful properties with high probability. They are "sampleable" and do not have too many problematic jobs clustered at the end of the sequence. A second step formalizes this, introducing *stable sequences*. Non-stable sequences are rare and negligible, we are thus reduced to stable sequences. The third step is a classical adversarial analysis that uses the properties of stable sequences to again establish worst-case guarantees.

The paper concludes with lower bounds. We show that no algorithm, deterministic or randomized, is better than nearly 1.257-competitive. This immediately implies a lower bound of 1.043 in the general secretary model.

Notation. We use the notation $[\mathcal{J}]$ or $[\mathcal{J}^\sigma]$ to highlight values that depend on the job set $\mathcal{J}$ or the ordered job sequence $\mathcal{J}^\sigma$. Such appendage is omitted when the dependency needs not be highlighted. In similar vein, we may write OPT for OPT$(\mathcal{J})$.

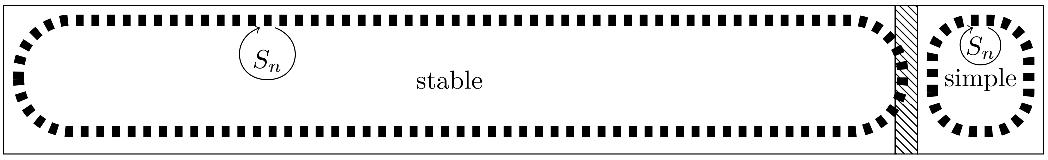

Figure 1 The lay of the land in our analysis. The algorithm is $(c+\varepsilon)$-competitive on simple and stable sequences. Only the small unstable remainder (hashed) is problematic. Dashed lines mark orbits under the action of the permutation group S_n. Simple sequences stay simple under permutation.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

2 A strong measure of random-order competitiveness

Consider a job set $\mathcal{J} = \{J_1, \dots, J_n\}$ of known size n. Each job is fully defined[1] by its non-negative size (or processing time) $p_1, \dots, p_n$. Let S_n be the group of permutations of the integers from 1 to n, which we consider a probability space under the uniform distribution. We pick each permutation with probability $1/n!$. Each permutation $\sigma \in S_n$, called an *order*, gives us a *job sequence* $\mathcal{J}^\sigma = J_{\sigma(1)}, \dots, J_{\sigma(n)}$. 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 random job order, i.e. $\mathbf{E}_{\sigma\sim S_n}[A(\mathcal{J}^\sigma)] = \frac{1}{n!}\sum_{\sigma \in S_n} A(\mathcal{J}^\sigma)$. We use the term *secretary model*, to distinguish this setting from the *random-order model* in [6] where the input size n is not known in advance. The algorithm A is *c-competitive in the secretary model* if we have $\mathbf{E}_{\sigma\sim S_n}[A(\mathcal{J}^\sigma)] \leq c\mathrm{OPT}(\mathcal{J})$ for all input sets $\mathcal{J}$.

The secretary 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* [6]. 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 sets $\mathcal{J}$ there holds $\mathbf{P}_{\sigma\sim S_n}[A(\mathcal{J}^\sigma) \geq (c+\varepsilon)OPT(\mathcal{J})] \leq \varepsilon$.*

▶ **Lemma 2.** *If a deterministic online algorithm is nearly c-competitive, then it is c-competitive in the secretary model for $m \to \infty$, i.e. for its competitive ratio c_m on m machines holds $\lim\limits_{m\to\infty} c_m = c$.*

3 Basic properties

Let us fix an input set $\mathcal{J}$. Graham [31] establishes that his greedy strategy is 2-competitive. He considers the *average load* $L = L[\mathcal{J}] = \frac{1}{m}\sum_{i=1}^m p_i$, which is the same for any schedule of the jobs in $\mathcal{J}$, and the maximum size of any job $p_{\max} = \max_i p_i$. Both are lower bounds for OPT. Indeed, even the best schedule cannot have all machines loads below average, i.e. smaller than L, and the machine containing the largest job has load at least $p_{\max}$. Now, Graham observes that the smallest load in any schedule cannot exceed the average load L. Greedily using the least loaded machine causes makespan at most $L + p_{\max} \leq 2\mathrm{OPT}$. The greedy strategy is thus 2-competitive.

Graham's argument builds the foundation for subsequent work on scheduling problems. The following proposition guarantees a (small) constant adversarial ratio for almost every sensible random-order algorithm, which is necessary for obtaining nearly competitiveness.

[1] We propose for completeness that jobs of similar size are indistinguishable. A unique identification, say the index or a hash value, could in theory be used to derandomize a randomized algorithm. All of the results in this paper hold independently of whether such identification is possible.

▶ **Proposition 3.** *Assume job J is scheduled on a machine M such that at most $i-1$ machines have strictly smaller load than M. Then load of M is at most $\left(\frac{m}{m-i}+1\right) OPT$ afterwards.*

Proof. Let l be the load of M prior to receiving job J. By assumption at least $m-i$ machines have load l. Thus $L \geq \frac{m-i}{m} l$. We schedule job J of size at most OPT on machine M of load at most $l \leq \frac{m}{m-i} L \leq \frac{m}{m-i}\mathrm{OPT}$. The resulting load is at most $\left(\frac{m}{m-i}+1\right) OPT$. ◀

The previous result cannot be improved in general. The most difficult adversarial sequences have $L \approx p_{\max} \approx \mathrm{OPT}$. Random-order arrival faces further challenges. Certain degenerate sequences, where few jobs carry all the load, are not suited for reordering arguments. See Figure 2. This "degeneracy" is measured by $R(\mathcal{J}) = \min(\frac{L}{p_{\max}}, 1)$. Adapting the previous arguments we obtain the following result, which indicates good performance in almost all situations if $R(\mathcal{J})$ is small.

▶ **Proposition 4.** *Let M be a machine such that at most $i-1$ machines have strictly smaller load than M. If M receives a job, its load is at most $\left(\frac{m}{m-i}R(\mathcal{J})+1\right) OPT$ afterwards.*

Proof. Adapt the previous proof using that $L \leq R(\mathcal{J})\mathrm{OPT}$. ◀

Proposition 3 and 4 form the basis of our analyses. They give conditions when to use the Least-Loaded-Strategy in the main algorithm, establish most of our worst-case guarantees and explain why we can exclude simple sequences like the one in Figure 2. In the full paper, we generalize these propositions further, which is required for the main algorithm.

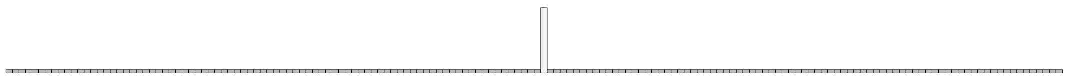

Figure 2 A surprisingly difficult sequence for random-order arguments. The big job carries most of the processing volume. Other jobs are negligible. Thus, all permutations look basically the same. Such "simple" job sets need to be excluded before the main analysis.

3.1 Sampling for Scheduling Problems

We now explain how we use sampling in the secretary model. Consider any input permutation $\mathcal{J}^\sigma = J_{\sigma(1)} \ldots J_{\sigma(n)}$. A standard technique is to sample the φ-fraction of jobs, $J_{\sigma(1)} \ldots J_{\sigma(\lceil \varphi n \rceil)}$, to make predictions about $\mathcal{J}^\sigma$. The previous section gives two prime candidates for sampling which relate to OPT, namely L and $p_{\max}$. Directly "sampling" OPT is futile.

The size $p_{\max}$ is best estimated by $p_{\max}^{\varphi t} = \max(p_{\sigma(t')} \mid \sigma(t') < \varphi n + 1)$. This corresponds to how we try to estimate the best secretary in the secretary-problem. Of course, $p_{\max}^{\varphi t}$ may vastly underestimate $p_{\max}$. If the sequence contains only a single huge job, this job is unlikely to be observed in the sample. Still, only very few jobs can have size exceeding $p_{\max}^{\varphi t}$ on random-order sequences; only $1/\varphi$ in expectation. The main algorithm uses reserve machines to catch these "exceptional" jobs.

For L we can get an unbiased[2] estimator from the sample: $L_\varphi = \frac{1}{\varphi m}\sum_{\sigma(t)\leq \varphi n} p_i$. Of course, we still need to determine how close L_φ is to L. Can we say that with high probability $L_\varphi \approx L$? For the sequence in Figure 2 such a statement cannot be true. The main observation

[2] The estimator is unbiased, i.e. $E[L_\varphi] = L$, if φn is a natural number. For general n, we could have replaced the factor $\frac{1}{\varphi m}$ in the definition of L_φ by the more complicated expression $\frac{n}{\lceil \varphi n \rceil m}$.

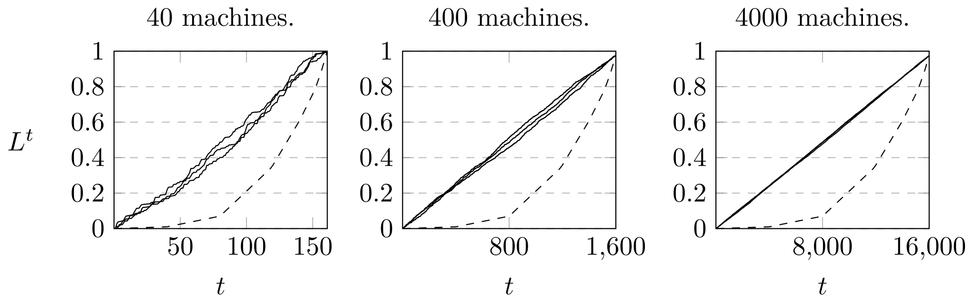

Figure 3 A graphic depicting the average load over time on the classical lower bound sequence from [1] for 40, 400 and 4000 machines. The dashed line corresponds to the original adversarial order. The three solid lines, corresponding to random permutations, clearly approximate a straight line. Thus, sampling allows to predict the (final) average load.

is that these counterexamples tend to have a small value $R(\mathcal{J})$. Given a lower bound $R_{\text{low}} > 0$ on $R(\mathcal{J})$ the following Load Lemma establishes $L_\varphi \approx L$. We have seen in the previous section that sequences with $R(\mathcal{J}) < R_{\text{low}}$ pose no major obstruction. The results in the previous section guarantee arbitrarily good performance if we choose $R_{\text{low}} > 0$ small enough.

The Load Lemma is quite potent and thus fundamental to random-order makespan minimization. It may be somewhat surprising to researchers on related problems since it makes implicit use of having non-small input sizes. Note that for our problem small inputs of size less than m are trivially scheduled optimally.

▶ **Lemma 5** (Load Lemma [6]). *Let $R_{\text{low}} = R_{\text{low}}(m) > 0$, $1 \geq \varphi = \varphi(m) > 0$ and $\varepsilon = \varepsilon(m) > 0$ be three functions in m such that $\varepsilon^{-4}\varphi^{-1}R_{\text{low}}^{-1} = o(m)$. Then there exists a variable $m(R_{\text{low}}, \varphi, \varepsilon)$, depending on these three functions, such that for $m \geq m(R_{\text{low}}, \varphi, \varepsilon)$ machines and all job sets $\mathcal{J}$ with $R(\mathcal{J}) \geq R_{\text{low}}$ and $|\mathcal{J}| \geq m$:*

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

A less general version of the Load Lemma already appeared in [6]. While the Load Lemma gives only asymptotic guarantees simulations show that it requires not very large numbers of machines. Figure 4 shows the expected value of $\left|\frac{L_{1/4}[\mathcal{J}^\sigma]}{L[\mathcal{J}]} - 1\right|$ on a suitable benchmark sequence.

For our more sophisticated algorithm we also use sampling to estimate the size of critical jobs. Consider a job class $\mathcal{C}$ of size $n_{\mathcal{C}} \in O(m)$. A consequence of Chebyshev's inequality, detailed in the full version, shows that we can estimate $n_{\mathcal{C}}$ up to an additive summand of $m^{3/4}$ after sampling a $\frac{1}{\log(m)}$-fraction of the sequence. In fact the load lemma is proven by sampling job classes obtained through geometric rounding.

4 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 $0 \leq t \leq n$, 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.

Let $\delta = \delta(m)$ be a certain *margin of error* our algorithm allows. It is optimal to set $\delta = 0$ but then the analysis requires a generalization of the result in [4]. In order for our main result to be self-contained one may set $\delta = \frac{1}{\log(m)}$, which allows to use results from [4] as a black box.

Given an input sequence $\mathcal{J}^\sigma$ we know from Section 3 that $\hat{L}_{\text{pre}} = \hat{L}_{\text{pre}}[\mathcal{J}^\sigma] = \frac{L_{1/4}[\mathcal{J}^\sigma]}{1-\delta}$ provides a good estimate of the (final) average load $L = \frac{1}{m}\sum_{i=1}^m p_i$. We use the index "pre" since our main algorithm later will use a slightly different guess $\hat{L}$. Consider the following adaptation LightLoadROM of the algorithm LightLoad from Albers and Hellwig [4].

Algorithm 1 The algorithm LightLoadROM.

```
1: Let $J_t$ be the job to be scheduled and let $p_t$ be its size.
2: if $t < n/4$ or $l_{\text{low}}^{t-1} \leq 0.25\hat{L}_{\text{pre}}$ or $l_{\text{mid}}^{t-1} + p_t > 1.75\hat{L}_{\text{pre}}$ then
3:     Schedule $J_t$ on any least loaded machine;
4: else schedule $J_t$ on $M_{\text{mid}}^{t-1}$;
```

▶ Remark 6. The first condition in the **if**-statement, $t < n/4$, already implies $l_{\text{low}}^{t-1} \leq 0.25\hat{L}_{\text{pre}}$ and is thus technically superfluous. We added it to clarify that LightLoadROM can be implemented as an online algorithm and only needs to know $\hat{L}_{\text{pre}}$ once $t \geq n/4$.

If we replace $\hat{L}_{\text{pre}}$ in the previous pseudocode by the average load L, we recover the semi-online algorithm LightLoad for makespan minimization, which has been analyzed by Albers and Hellwig [4]. They show that the algorithm is 1.75-competitive for $L = \hat{L}_{\text{pre}}$. We can show that the algorithm can also be used for general values $\hat{L} \approx L$. The performance gracefully decreases with $|L - \hat{L}_{\text{pre}}|$.

▶ **Theorem 7.** *Let $\mathcal{J}^\sigma$ be any (ordered) input sequence. The makespan of* LightLoadROM *on $\mathcal{J}^\sigma$ is at most* $1.75\left(1 + \frac{|\hat{L}_{\text{pre}}[\mathcal{J}^\sigma] - L|}{L}\right)\text{OPT}$.

Proof Sketch. For $\hat{L}_{\text{pre}} = L$, this is the main result in [4].

ItFor $\hat{L}_{\text{pre}} \geq L$, we can reduce ourselves to the case $\hat{L}_{\text{pre}} = L$. Consider any machine M in the optimum schedule of $\mathcal{J}$ that has load $l_M < \max(\hat{L}_{\text{pre}}, \text{OPT})$. We assign an additional job J_M of size $p_M = \max(\hat{L}_{\text{pre}}, \text{OPT}(\mathcal{J})) - l_M$ to this machine. For the resulting job set $\mathcal{J}'$ clearly $\text{OPT}(\mathcal{J}') = L(\mathcal{J}') = \max(\hat{L}_{\text{pre}}, \text{OPT})$. We can apply the main result of [4] to see that LightLoad has makespan at most $1.75\max(\hat{L}_{\text{pre}}, \text{OPT}(\mathcal{J}))$ if it first schedules the jobs $\mathcal{J}^\sigma$ (in order σ) followed by the additional jobs. But on the prefix $\mathcal{J}^\sigma$ LightLoad behaves precisely like LightLoadROM on input $\mathcal{J}^\sigma$. Thus, LightLoadROM has makespan at most $1.75\max(\hat{L}_{\text{pre}}[\mathcal{J}^\sigma], \text{OPT}(\mathcal{J}))$. Then, the theorem follows for $\hat{L}_{\text{pre}} \geq L$ since $\hat{L}_{\text{pre}} \leq (1 + \frac{\hat{L}_{\text{pre}} - L}{L})L \leq (1 + \frac{|\hat{L}_{\text{pre}}[\mathcal{J}^\sigma] - L|}{L})\text{OPT}$.

If $\hat{L}_{\text{pre}} \leq L$, the statement of the theorem still holds. can be derived similar to the analysis in [4]. Unfortunately, it cannot be immediately deduced from their results. Instead, their proofs need to be adapted. We sketch the necessary adaptations in the full version. ◀

The previous theorem already establishes a constant adversarial competitive ratio of 7. Use that $0 \leq \hat{L}_{\text{pre}} \leq L_{1/4} \leq 4L$ implies $|\hat{L}_{\text{pre}}[\mathcal{J}^\sigma] - L| \leq 3L$. We can improve this result, most importantly, if $R(\mathcal{J})$ is small.

▶ **Lemma 8.** *For any (ordered) job sequence $\mathcal{J}^\sigma$ the makespan of* LightLoadROM *is at most $(1 + 2R(\mathcal{J}))\text{OPT}(\mathcal{J})$. In particular, it is at most $3\,\text{OPT}(\mathcal{J})$ in general and at most $1.75\,\text{OPT}(\mathcal{J})$ for $R(\mathcal{J}) < 3/8$.*

Proof. Since LightLoadROM only considers the least or the $\lfloor m/2 \rfloor$-th least loaded machine, the lemma follows from Proposition 4. ◀

We now establish the competitive ratio of LightLoadROM in the strong model of nearly competitiveness. Corollary 10 follows immediately by Lemma 2.

▶ **Theorem 9.** *The algorithm* LightLoadROM *is nearly* 1.75*-competitive.*

▶ **Corollary 10.** LightLoadROM *is* 1.75*-competitive in the secretary model for* $m \to \infty$*.*

Proof of Theorem 9. Our analysis forms a triad outlining how we analyze the more sophisticated 1.535-competitive main algorithm. See Figure 1 for an illustration. Since we only prove the case $\hat{L} \geq L$ of Theorem 7, we will not rely on the case $L \leq \hat{L}$ in this proof. For this, we need to set $\delta(m) = \frac{1}{\log(m)}$.

Analysis basics. By Lemma 8 algorithm LightLoadROM is 3-competitive in the adversarial model. The first condition of nearly competitiveness is satisfied. We call input set $\mathcal{J}$ *simple* if $|\mathcal{J}| \leq m$ or $R[\mathcal{J}] < \frac{3}{8}$. Observe that LightLoadROM is (adversarially) 1.75-competitive on simple job sets. Indeed, if $|\mathcal{J}| < m$ LightLoadROM assigns every job to an empty least-loaded machine, which is obviously optimal. If $R[\mathcal{J}] < \frac{3}{8}$, Lemma 8 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 $L \leq \hat{L}_{\text{pre}} \leq \frac{1+\delta(m)}{1-\delta(m)} L$. If a sequence is proper, it fulfills the conditions of the Load Lemma with $\varphi = 1/4$, $R_{\text{low}} = \frac{3}{8}$ and $\varepsilon(m) = \delta(m) = 1/\log(m) \in \omega(m^{-1/4})$. The Load Lemma guarantees that for m large enough, $\mathbf{P}_{\sigma \sim S_n}\left[\left|\frac{L_\varphi[\mathcal{J}^\sigma]}{L[\mathcal{J}]} - 1\right| \geq \delta\right] < \delta$. Note that $\left|\frac{L_\varphi[\mathcal{J}^\sigma]}{L[\mathcal{J}]} - 1\right| < \delta$ is equivalent to $(1-\delta)L < L_\varphi[\mathcal{J}^\sigma] < (1+\delta)L$, which in turn implies that $L \leq \hat{L}_{\text{pre}} \leq \frac{1+\delta(m)}{1-\delta(m)} L$. Thus, the probability of the sequence $\mathcal{J}^\sigma$ being stable is at least $1 - \delta$ for m large enough and $\mathcal{J}$ proper.

Adversarial Analysis. By Theorem 7, the makespan of LightLoadROM on stable sequences with $L \leq \hat{L}_{\text{pre}} \leq \frac{1+\delta(m)}{1-\delta(m)} L$ is at most $1.75 \cdot \frac{1+\delta(m)}{1-\delta(m)} \text{OPT} = \left(1.75 + \frac{3.5 \cdot \delta(m)}{1-\delta(m)}\right) \text{OPT}(\mathcal{J})$. We only require the easy case, $L \leq \hat{L}_{\text{pre}}$ of Theorem 7, which is fully proven in this paper.

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 $\mathbf{P}_{\sigma \sim S_n}[\text{LightLoadROM}(\mathcal{J}^\sigma) \geq (1.75 + \varepsilon) OPT(\mathcal{J})] \leq \delta(m) \leq \varepsilon$ since the only sequences where the inequality does not hold are proper but not stable. This concludes the second condition of nearly competitivity.

The δ-term. Setting $\delta = 0$ can increase the $\frac{|\hat{L}_{\text{pre}}[\mathcal{J}^\sigma] - L|}{L}$-term in Theorem 7 by at most $1/\log(m)$, which vanishes for $m \to \infty$. Of course, in reality LightLoadROM improves for $\delta = 0$. ◀

Analyzing the algorithm LightLoadROM on small numbers of machines.

From now on, we consider LightLoadROM with $\delta = 0$. Thus, the average load L is estimated by $\hat{L}_{\text{pre}} = L_{1/4}$. The *normalized absolute mean deviation* of $\hat{L}_{\text{pre}} = L_{1/4}$ is defined as $\text{NMD}(\hat{L}_{\text{pre}}) = \mathbf{E}_{\sigma \sim S_n}\left[\frac{|\hat{L}_{\text{pre}}[\mathcal{J}^\sigma] - L|}{L}\right]$. The following is a consequence of Theorem 7 .

▶ **Theorem 11.** *On input set* $\mathcal{J}$ *the competitive ratio of* LightLoadROM *in the secretary model is at most* $1.75(1 + \text{NMD}(\hat{L}_{\text{pre}}))$.

In the full version we give an estimation on $\text{NMD}(\hat{L}_{\text{pre}})$, which leads to the following result.

▶ **Theorem 12.** *The competitive ratio of* LightLoadROM *is* $1.75 + \frac{18}{\sqrt{m}} + O\left(\frac{1}{m}\right)$.

The techniques presented in this section can, in theory, be extended to analyze the main algorithm in the next section. This is impractical due to the complexity of the analysis at hand. We are certain that the error term involved will be of the form $m^{-1/a}$ for a small.

The constant summand $\frac{18}{\sqrt{m}}$ in Theorem 12 is pessimistic. We discuss several avenues of further improvement in the full version. The best we are aware of allows for a competitive ratio as small as $\frac{4.4}{\sqrt{m}} + \frac{7}{m} + O\left(\frac{1}{m^{3/2}}\right)$ but there are ways to improve even further. The terms still hidden in the O-notation result from the Stirling-approximation and are known to be tiny. Figure 4 shows $\mathrm{NMD}(\hat{L}_{\mathrm{pre}})$ on the lower bound from [1], which is a sensible benchmark.

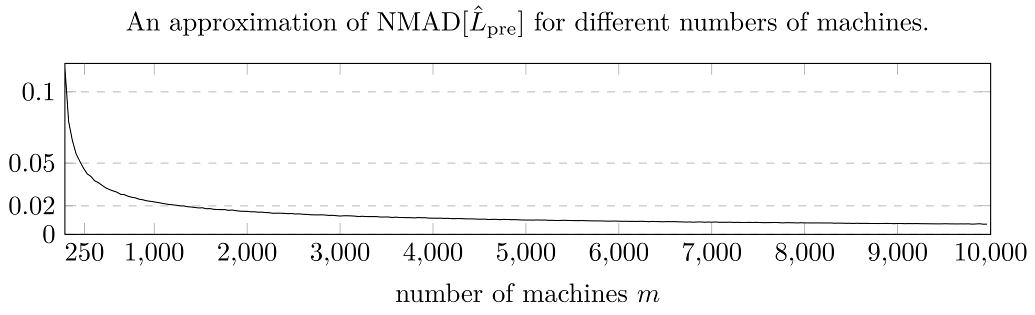

Figure 4 The extra cost for small numbers of machines. The graph shows an estimation of $\mathrm{NMD}(\hat{L}_{\mathrm{pre}})$ on the lower bound sequence from [1] based on $10,000$ random samples. Theorem 11 this indicates good performance of LightLoadROM in practice.

5 The nearly 1.535-competitive algorithm

The new main algorithm achieves a competitive ratio of $c = \frac{1+\sqrt{13}}{3} \approx 1.535$. It consists of three components: a sampling phase, the Least-Loaded-Strategy and the Critical-Job-Strategy. We now give a simplified description of the algorithm.

The sampling phase. A few jobs are sampled to predict the whole sequence. These Jobs are scheduled greedily with a some machines kept in reserve. This phase is uninformed and "mistakes" are unavoidable. Such mistakes are few, since the processing volume scheduled is small – at least if we exclude worst-case sequences. First, we sample B, which tries to estimate $\max(p_{\max}, L) \leq \mathrm{OPT}$. We then use sampling to predict *critical* jobs of size in between $(c-1)B$ and B. Intuitively, jobs smaller than $(c-1)B$ are too small to pose a problem. Jobs larger than B are also critical but cannot be predicted since they did not appear during sampling. This in turn means that they are rare. We keep a few reserve machines to safely process them.

The Critical-Job-Strategy. Our plan is to assign critical jobs ahead of time. Formally, placeholder jobs are used to reserve space for jobs yet to come. Critical jobs are assigned according to an easy heuristic: Each machine gets either one big or two medium jobs. Reserve machines handle errors in the predictions and unexpected huge jobs.

The Least-Loaded-Strategy. Sometimes the Critical-Job-Strategy is not feasible; there simply are too many critical jobs. This may already by apparent from sampling predictions, but for some job sets this cannot be predicted. The latter input sets form the crux of the analysis. Once we find out, we pick the Least-Loaded-Strategy, which enhances a Graham's greedy approach by still maintaining reserve machines for particularly large jobs. Intuitively, many critical jobs make it even for OPT impossible to schedule all jobs efficiently, which is why we rely on this less sophisticated strategy.

Further challenges. Algorithm design and analysis have to deal with three further issues. First, the Critical-Job-Strategy needs to take scheduling decisions made during sampling into account. Second, a consequence of sampling is that no value is exact, small sources of errors are imminent. Third, we need a constant competitive ratio against an adversary. All these challenges impact details of the algorithm design in rather subtle ways.

5.1 Formal Description

Let $\delta = \delta(m) = \frac{1}{\log(m)}$ be the *margin of error our algorithm allows.* Most of the time, it is sensible to treat δ as a constant and forget about its dependency on m. Our algorithm maintains a set of $\lceil \delta m \rceil$ *reserve machines.* Their complement are the *principal machines.* 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. This bound 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 $p_{\text{small}} = c - 1 = \frac{\sqrt{13}-2}{3} \approx 0.535$ and $p_{\text{big}} = \frac{c}{2} = \frac{1+\sqrt{13}}{6} \approx 0.768$. We call job J_t

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

Consider the set $\mathcal{P} = \{(1+\delta)^i \mid p_{\text{small}} B \leq (1+\delta)^i \leq B\}$ corresponding to rounded sizes of critical jobs. Given $p \in \mathcal{P}$ let n_p be the total number of p-jobs. We could estimate n_p by $\delta^{-2}\hat{n}_p$ after sampling where $\hat{n}_p = |\{J_{\sigma(j)} \mid \sigma(j) \leq \delta^2 n \wedge J_{\sigma(j)} \text{ is a } p\text{-job}\}|$ after sampling. In practice 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 two advantages. The value c_p is close to n_p with high probability, but unlikely to exceed it. Overestimating n_p turns out to be far worse than underestimating it. It also simplifies the description of the algorithm allowing medium jobs to always "pair up".

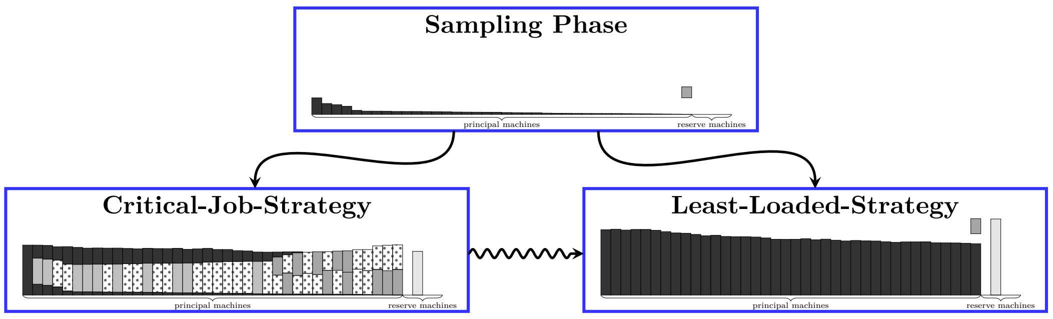

Figure 5 The 1.535-competitive algorithm. First, few jobs are sampled. Then, the algorithm decides between two strategies. The Critical-Job-Strategy tries to schedule critical jobs ahead of time. The Least-Loaded-Strategy follows a greedy approach, which reserves some machines for large jobs. Sometimes, 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.

Statement of the algorithm. If there are less jobs than machines, each job is placed onto a separate machine. This is optimal. Else, a short sampling phase greedily assigns each of the first $\lfloor \delta^2 n \rfloor$ jobs to the least loaded principal machine. Now, B and $(c_p)_{p \in \mathcal{P}}$ are known. We choose the Least-Loaded-Strategy if we predict the Critical-Job-Strategy to be infeasible. Formally, if $\sum_{p \in \mathcal{P}} w(p) c_p > m$, where $w(p) = 1/2$ for $p \leq p_{\text{big}}$ and $1 > p \leq p_{\text{big}}$. If $\sum_{p \in \mathcal{P}} w(p) c_p \leq m$, we choose the Critical-Job-Strategy. The Critical-Job-Strategy requires a one-time preparation. It may later switch to the Least-Loaded-Strategy but it never switches the other way around.

The Least-Loaded-Strategy

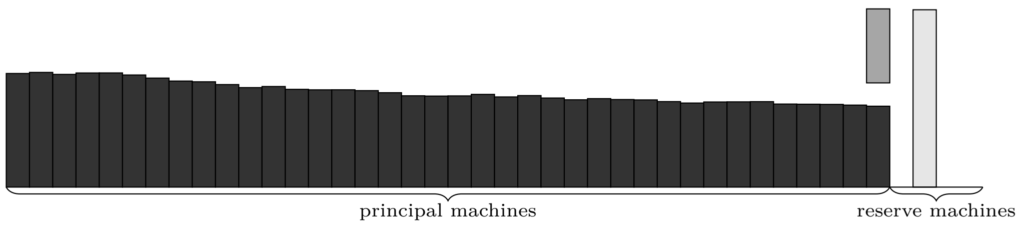

Figure 6 The Least-Loaded-Strategy schedules jobs greedily. A few machines are reserved for unexpected huge jobs. For example the largest job, which is unlikely to arrive in the sampling phase.

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

The Critical-Job-Strategy

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. They are assigned in the **Preparation for the Critical-Job-Strategy**. We try to pair off medium jobs, some of which already arrived during sampling. Moreover it is important to assign fewer processing volume to those machines, which have a higher load after the sampling phase.

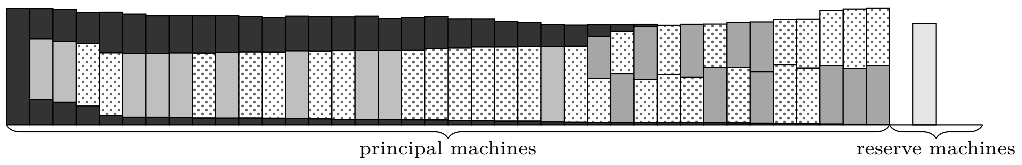

Figure 7 The Critical-Job-Strategy. Each machine gets either two medium, one large or no critical job. Placeholder jobs (dotted) reserve space for critical jobs yet to come. Processing volume of small jobs (dark) on the bottom arrived during the sampling phase. Reserve machines accommodate huge jobs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

The **Critical-Job-Strategy** places small jobs on least-loaded principal machines taking placeholders into account. Critical jobs of rounded size $p \in \mathcal{P}$ replace p-placeholder-jobs whenever possible. If no matching placeholder is found or if the current job is exceptional, the reserve machines are used. Again, medium jobs are paired up. If the reserve machines are full, the algorithm *fails*. It switches to the Least-Loaded-Strategy.

The full description of the main algorithm is provided in Appendix B.

6 Analysis of the algorithm

Theorem 13 is main result of the paper.

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

Due to Lemma 2 this competitive ratio also holds in the general secretary model.

▶ **Corollary 14.** *Our algorithm is c-competitive in the secretary model as $m \to \infty$.*

The analysis of the algorithm proceeds along the same three reduction steps used in the proof of Theorem 9. First, we assert that our algorithm has a constant 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 3.

▶ **Definition 15.** *A job set $\mathcal{J}$ is called* simple *if $R(\mathcal{J}) \le \frac{(1-\delta)\delta^2}{\delta^2+1}(c-1)$ 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.*

▶ **Main Lemma 16.** In the adversarial model our algorithm has competitive ratio $4 + O(\delta)$ on general input sequences and $c + O(\delta)$ on simple sequences.

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. Their formal definition can be found later in Definition 22. The second reduction shows that stable sequences arise with high probability if the order of a proper job set $\mathcal{J}$ is picked 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 17.** $\lim_{m\to\infty} P(m) = 0$.

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 stable sequences. This analysis is quite general. In particular, it does not rely further on random-order arrival. Instead, we work with worst-case stable inputs, i.e. we allow an adversary to present any stable input sequence.

▶ **Main Lemma 18.** Our algorithm is adversarially $(c+O(\delta))$-competitive on stable sequences.

These three main lemmas allow us to conclude the proof of Theorem 13.

Proof of Theorem 13. Recall that $\delta(m) \to 0$ for $m \to \infty$. By Main Lemma 16, the first condition of nearly competitiveness holds, i.e. our algorithm has a constant competitive ratio. Moreover, by Main Lemma 16 and Main Lemma 18, 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. This implies that for $m \ge m_0(\varepsilon)$ 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 input sequence is not stable. By Main Lemma 17, we can find $m(\varepsilon) \ge m_0(\varepsilon)$ such that $P(m) \le \varepsilon$ for $m \ge m(\varepsilon)$. This choice of $m(\varepsilon)$ satisfies the second condition of nearly competitiveness. ◀

Proof sketch of Main Lemma 16

The *anticipated load* $\tilde{l}_M^t$ of a machine M at time t denotes its load including placeholder-jobs. We can obtain the following two bounds on the average anticipated load $\tilde{L} = \sup_t \frac{1}{m} \sum_M \tilde{l}_M^t$.

▶ **Lemma 19.** *We have $\tilde{L} \le L + 2p_{\max}$, as well as $\tilde{L} \le \left(1 + \frac{1}{\delta^2}\right) L$.*

Thus the average anticipated load $\tilde{L}$ relates to the original values L, $p_{\max}$. In the full version, we generalize Proposition 4 to anticipated loads. We can then use Lemma 19 to conclude Main Lemma 16. The only exception are reserve machines, which receives its last job using the Critical-Job-Strategy. Their load needs to be bounded using different techniques.

Formally, we can prove the following two statements, which imply Main Lemma 16.

▶ **Proposition 20.** *The main algorithm is adversarially $\left(1 + \frac{3}{1-\delta} + 2\delta\right)$-competitive.*

▶ **Proposition 21.** *The main algorithm has makespan at most $(c + 2\delta)\mathrm{OPT}$ on simple sequences $\mathcal{J}^\sigma$.*

Stable job sequences and a proof sketch of Main Lemma 17

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 exceptional 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 22.** *A job sequence $\mathcal{J}^\sigma$ is* stable *if the following conditions hold:*

- *The estimate $\hat{L}$ for L is accurate, i.e. $(1-\delta)L \le \hat{L} \le (1+\delta)L$.*
- *The estimate c_p for n_p is accurate, i.e. $c_p \le n_p \le c_p + 2m^{3/4}$ for all $p \in \mathcal{P}$.*
- *There are at most $\lceil \delta m \rceil$ exceptional jobs in $\mathcal{J}^\sigma$.*
- *Let $\tilde{t}$ be the time the last exceptional 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 $n_{p,\tilde{t}} \le \left(1 - \delta^3\right) n_p$ for every $p \in \mathcal{P}$.*
- *$\delta^3 \left\lfloor \left(1 - \delta - 2\delta^2\right) m/|\mathcal{P}| \right\rfloor \ge 2|\mathcal{P}|m^{3/4}$.*

Proof sketch of Main Lemma 17. The first condition follows from Lemma 5. The second condition can be derived using Chebyshev's inequality as discussed at the end of Section 3.1. Both conditions 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. In expectation the rank of the largest job occurring in the sampling phase is δ^{-2}, a constant. The fourth condition states that the exceptional jobs are evenly spread throughout the sequence compared to the p-jobs for any $p \in \mathcal{P}$. Again, this is expected of a random sequence and corresponds to how one would determine randomness statistically. For the final condition it suffices to choose the number of machines m large enough. One technical problem arises since the set of rounded critical job sizes $\mathcal{P} = \mathcal{P}[\mathcal{J}^\sigma]$ 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. ◀

Proof sketch of Main Lemma 18

We first consider the Critical-Job-Strategy. Main Lemma 18 holds as long as it is employed.

▶ **Lemma 23.** *The makespan of our algorithm is at most $(c + O(\delta)) \max(B, L, p_{\max})$ on stable sequences till it employs the Least-Loaded-Strategy (or till the end of the sequence).*

Proof sketch. Let us only consider critical jobs at any time the Least-Loaded-Strategy is not employed. We can then show that a machine contains either one big job or at most two medium jobs. In the first case we bound the size of this big, possibly exceptional, job by $p_{\max}$. Else, if the machine contains two medium jobs their total weight 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 caused by small jobs requires techniques similar to the proof of Main Lemma 16. ◀

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 corollary to Lemma 23.

▶ **Corollary 24.** *Till our algorithm uses the Least-Loaded-Strategy its makespan is less than* $(c + O(\delta))\text{OPT}$ *on stable sequences.*

Hence, we are left to consider the Least-Loaded-Strategy. We say the algorithm *fails* if it has to switch from the Critical-Job-Strategy to the Least-Loaded-Strategy. The following lemma is crucial and relies deeply on the properties of stable sequences, in particular the fourth one.

▶ **Lemma 25.** *If the algorithm fails, every exceptional job has been scheduled.*

The lemma shows that the Least-Loaded-Strategy only needs to deal with exceptional jobs if it is picked immediately. In this case, all reserve machines are empty. The third property of stable sequences ensures that there are enough reserve machines so that every exceptional job is assigned to an empty machine.

Non-exceptional jobs, i.e. jobs of size at most B, are scheduled onto a least loaded principal machine. This machine was among the $\delta m + 1$ least loaded machines and had load at most $mL/(m - \delta m + 1)$ by Proposition 4. Afterwards, its load was at most $mL/(m - \delta m + 1) + B \le (2 + O(\delta))B$ since $(1 - \delta)L \le B$ for stable sequences. The following lemma concludes the proof of Main Lemma 18 since it implies that $(2 + O(\delta))B \le (c + O(\delta))\text{OPT}$.

▶ **Lemma 26.** *If the Least-Loaded-Strategy is applied on a stable sequence,* $B \le \frac{c}{2}\text{OPT}$.

The proof of Lemma 26 is left to the full version.

7 Lower bounds

We establish the following theorem using two lower bound sequences. These results generalize to randomized algorithms using appropriate notions of (nearly) competitiveness.

▶ **Theorem 27.** *For every online algorithm A, deterministic or randomized, there exists a job set $\mathcal{J}$ such that* $\mathbf{P}_{\sigma \sim S_n}\left[A(\mathcal{J}^\sigma) \ge \frac{\sqrt{73}-1}{6}\text{OPT}(\mathcal{J})\right] \ge \frac{1}{6}$. *If A is randomized the previous probability also includes its random choices.*

The lower bound highlights the inability of the main algorithm to decide between the Least-Loaded-Strategy and the Critical-Job-Strategy. If we could communicate this decision, say through a single advice bit, our main algorithm would become nearly optimal, i.e. nearly 1-competitive, on the lower bound sets. Theorem 27 implies the following lower bounds.

▶ **Corollary 28.** *If an online algorithm A is nearly c-competitive, then* $c \ge \frac{\sqrt{73}-1}{6} \approx 1.257$.

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

The lower bounds are proven in the appendix.

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, W. Gálvez, and M. Janke. Machine covering in the random-order model. In *32nd International Symposium on Algorithms and Computation (ISAAC 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

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

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

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

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

8 Pablo Azar, Robert Kleinberg, and S. Weinberg. Prophet inequalities with limited information. *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, July 2013. `doi:10.1137/1.9781611973402.100`.

9 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.

10 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.

11 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.

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

13 Martin Böhm, Jiří Sgall, Rob Van Stee, and Pavel Veselỳ. A two-phase algorithm for bin stretching with stretching factor 1.5. *Journal of Combinatorial Optimization*, 34(3):810–828, 2017.

14 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.

15 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.

16 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.

17 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.

18 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.

19 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.

20 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.

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

22 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.

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

24 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.

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

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

27 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.

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

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

30 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.

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

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

33 A. Gupta and S. Singla. Random-order models. In Tim Roughgarden, editor, *Beyond worst-case analysis*. Cambridge University Press, 2020.

34 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.

35 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.

36 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.

37 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.

38 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.

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

40 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.

41 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.

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

43 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.

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

45 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.

46 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.

47 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.

48 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.

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

50 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.

51 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.

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

53 J. Rudin III. *Improved bounds for the on-line scheduling problem.* PhD thesis, University of Phoenix, 2001.

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

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

A Lower bounds

We establish the following theorem using two lower bound sequences.

▶ **Theorem 27.** *For every online algorithm A, deterministic or randomized, there exists a job set $\mathcal{J}$ such that* $\mathbf{P}_{\sigma\sim S_n}\left[A(\mathcal{J}^\sigma) \geq \frac{\sqrt{73}-1}{6}\mathrm{OPT}(\mathcal{J})\right] \geq \frac{1}{6}$. *If A is randomized the previous probability also includes its random choices.*

Theorem 27 implies the following lower bounds.

▶ **Corollary 28.** *If an online algorithm A is nearly c-competitive, then* $c \geq \frac{\sqrt{73}-1}{6} \approx 1.257$.

▶ **Corollary 29.** *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.

▶ **Lemma 30.** *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 31.** *For job set $\mathcal{J}_1$ we have $\mathbf{P}_{\sigma\sim S_n}[A(\mathcal{J}_1^\sigma) \geq c\mathrm{OPT}(\mathcal{J})] \geq \frac{1-P}{3}$ and for job set $\mathcal{J}_2$ furthermore $\mathbf{P}_{\sigma\sim S_n}[A(\mathcal{J}_2^\sigma) \geq c\mathrm{OPT}(\mathcal{J})] \geq \frac{P}{3}$.*

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 $\mathrm{OPT}(\mathcal{J}_1) = 1$. Thus with probability $\frac{1-P}{3}$ we have $A(\mathcal{J}_1^\sigma) = c = c\mathrm{OPT}(\mathcal{J}_1)$.

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 $\mathrm{OPT}(\mathcal{J}_2) = c$. In particular we have with probability $\frac{P}{3}$ that $A(\mathcal{J}_2^\sigma) = c^2 = c\mathrm{OPT}(\mathcal{J}_2)$. ◀

We now conclude the main three lower bound results.

▶ **Theorem 27.** *For every online algorithm A, deterministic or randomized, there exists a job set $\mathcal{J}$ such that $\mathbf{P}_{\sigma\sim S_n}\left[A(\mathcal{J}^\sigma) \geq \frac{\sqrt{73}-1}{6}\mathrm{OPT}(\mathcal{J})\right] \geq \frac{1}{6}$. If A is randomized the previous probability also includes its random choices.*

Proof. By the previous lemma we get that

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

▶ **Corollary 28.** *If an online algorithm A is nearly c-competitive, then $c \geq \frac{\sqrt{73}-1}{6} \approx 1.257$.*

Proof. This is immediate by the previous theorem. ◀

▶ **Corollary 29.** *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 27. Then

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

B Full description of the main 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 $\mathcal{M}_{\text{res}}$ 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. This bound B 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 $p_{\text{small}} = c - 1 = \frac{\sqrt{13}-2}{3} \approx 0.535$ and $p_{\text{big}} = \frac{c}{2} = \frac{1+\sqrt{13}}{6} \approx 0.768$. Then we call job J_t

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

Consider the sets $\mathcal{P}_{\text{med}} = \{(1+\delta)^i \mid (1+\delta)^{-1} p_{\text{small}} B < (1+\delta)^i \leq p_{\text{big}} B\}$ and $\mathcal{P}_{\text{big}} = \{(1+\delta)^i \mid p_{\text{big}} B < (1+\delta)^i \leq B\}$ corresponding to all possible rounded sizes of medium respectively big jobs, excluding huge jobs. Let $\mathcal{P} = \mathcal{P}_{\text{med}} \cup \mathcal{P}_{\text{big}}$. This subdivision gives rise to a *weight function*, which will be important later. Let $w(p) = 1/2$ for $p \in \mathcal{P}_{\text{med}}$ and $w(p) = 1$ for $p \in \mathcal{P}_{\text{big}}$. 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. We let $n_p = |\mathcal{C}_p|$ and $\hat{n}_p = |\{J_{\sigma(j)} \mid \sigma(j) \leq \delta^2 n \wedge J_{\sigma(j)} \text{ is a } p\text{-job}\}|$. 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 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.

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 ← 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);
```

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.

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.

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 3.

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 $c'_p \le c_p$. 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 32 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 32. 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.

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 $p \in \mathcal{P}_{\text{med}}$ to $M$; $c'_p \leftarrow c'_p + 1$;
3: while there is an unsaturated size class $p \in \mathcal{P}$ and $|\mathcal{J}_{\text{rep}}| < m_{\text{empty}}$ 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 $q \in \mathcal{P}_{\text{med}}$ unsaturated. $e \leftarrow (p,q)$; $w(e) \leftarrow p+q$; $c'_q \leftarrow c'_q + 1$;
6:     Add $e$ to $\mathcal{J}_{\text{rep}}$;
7: while $\mathcal{J}_{\text{rep}} \neq \emptyset$ do
8:     Pick a least loaded machine $M \in \mathcal{M}$, which does not contain a critical job yet;
9:     Pick $e \in \mathcal{J}_{\text{rep}}$ of maximum size $w(e)$ and add the jobs in $e$ to $M$;
10:    $\mathcal{J}_{\text{rep}} \leftarrow \mathcal{J}_{\text{rep}} \setminus \{e\}$;
```

▶ **Lemma 32.** *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 $\sum_{p\in\mathcal{P}_{\text{med}}}\hat{n}_p$ machines with critical jobs while there are at least $\sum_{p\in\mathcal{P}_{\text{med}}}(c_p-\hat{n}_p)\geq\sum_{p\in\mathcal{P}_{\text{med}}}\hat{n}_p$ placeholder-jobs available to fill them. Here we make use of the fact that for medium jobs $p\in\mathcal{P}_{\text{med}}$ 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 $\mathcal{J}_{\text{rep}}$ contains an even number of medium jobs. If the placeholder picked in line 4 was the last medium job remaining, $\sum_{p\in\mathcal{P}_{\text{med}}}c_p$ would be odd. But this is not the case since every c_p for $p\in\mathcal{P}_{\text{med}}$ is even. ◀

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 $M\in\mathcal{M}_{\text{res}}$ containing a single medium job then
5:         Schedule $J_t$ on $M$;
6:     else if there exists an empty machine $M\in\mathcal{M}_{\text{res}}$ then schedule $J_t$ on $M$;
7:     else STAT ← 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);
```

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.