Machine Covering in the Random-Order Model

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

Waldo Gálvez ✉ 🏠 ⓘ
Institute of Engineering Sciences, Universidad de O'Higgins, Rancagua, Chile

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

─── **Abstract** ───

In the Online Machine Covering problem jobs, defined by their sizes, arrive one by one and have to be assigned to m parallel and identical machines, with the goal of maximizing the load of the least-loaded machine. Unfortunately, the classical model allows only fairly pessimistic performance guarantees: The best possible deterministic ratio of m is achieved by the Greedy-strategy, and the best known randomized algorithm has competitive ratio $\tilde{O}(\sqrt{m})$ which cannot be improved by more than a logarithmic factor.

Modern results try to mitigate this by studying semi-online models, where additional information about the job sequence is revealed in advance or extra resources are provided to the online algorithm. In this work we study the Machine Covering problem in the recently popular *random-order* model. Here no extra resources are present, but instead the adversary is weakened in that it can only decide upon the input set while jobs are revealed uniformly at random. It is particularly relevant to Machine Covering where lower bounds are usually associated to highly structured input sequences.

We first analyze Graham's Greedy-strategy in this context and establish that its competitive ratio decreases slightly to $\Theta\left(\frac{m}{\log(m)}\right)$ which is asymptotically tight. Then, as our main result, we present an improved $\tilde{O}(\sqrt[4]{m})$-competitive algorithm for the problem. This result is achieved by exploiting the extra information coming from the random order of the jobs, using sampling techniques to devise an improved mechanism to distinguish jobs that are relatively large from small ones. We complement this result with a first lower bound showing that no algorithm can have a competitive ratio of $O\left(\frac{\log(m)}{\log\log(m)}\right)$ in the random-order model. This lower bound is achieved by studying a novel variant of the Secretary problem, which could be of independent interest.

1 Introduction

We study the *Machine Covering* problem, a fundamental load balancing problem where n jobs have to be assigned (or scheduled) onto m identical parallel machines. Each job is characterized by a non-negative size, and the goal is to maximize the smallest machine load. This setting is motivated by applications where machines consume resources in order to work, and the goal is to keep the whole system active for as long as possible. Machine Covering has found additional applications in the sequencing of maintenance actions for

aircraft engines [20] and in the design of robust Storage Area Networks [41]. The offline problem, also known as Santa-Claus or Max-Min Allocation Problem, received quite some research interest, see [44, 10, 6] and references therein. In particular, the problem is known to be strongly NP-hard but to allow for a Polynomial-Time Approximation Scheme (PTAS) [44].

This paper focuses on the online version of the problem, where jobs arrive one by one and must be assigned to some machine upon arrival. Lack of knowledge about future jobs can enforce very bad decisions in terms of the quality of the constructed solutions: In a classical lower bound sequence, m jobs of size 1 arrive first to the system and they must be assigned to m different machines by a competitive deterministic online algorithm. Then subsequent $m - 1$ jobs of size m arrive, which make the online algorithm perform poorly, see Figure 1. Indeed, the best possible deterministic algorithm achieves a competitive ratio of m [44], and if randomization is allowed, the best known competitive ratio is $\tilde{O}(\sqrt{m})$, which is best possible up to logarithmic factors [7]. The corresponding lower bound also uses that the online algorithm cannot schedule the first m jobs correctly, at least not with probability exceeding $\frac{1}{\sqrt{m}}$.
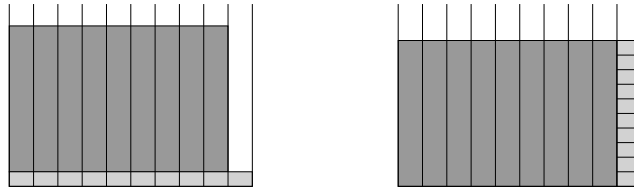
■ **Figure 1** The instance showing that no deterministic algorithm is better than m-competitive for Machine Covering. To the left, the best possible solution that an online algorithm can construct achieves minimum load 1. To the right, the optimal minimum load is m.

Such restrictive facts have motivated the study of different semi-online models that provide extra information [7, 14, 34, 37] or extra features [41, 42, 22, 16] to the online algorithm.

This work studies the Online Machine Covering problem in the increasingly popular *random-order* model. In this model, jobs are still chosen worst possible by the adversary but they are presented to the online algorithm in a uniformly random order. The random-order model derives from the Secretary Problem [13, 35] and has been applied to a wide variety of problems such as generalized Secretary problems [32, 8, 33, 18, 9], Scheduling problems [39, 38, 2, 3], Packing problems [30, 31, 5, 4], Facility Location problems [36] and Convex Optimization problems [25] among others. See also [26] for a survey chapter. It is particularly relevant to Machine Covering, where hard instances force online algorithms to make an irredeemable mistake right on the first m jobs due to some hidden large job class at the end.

Indeed, we show that the competitive ratio of Graham's Greedy-strategy improves from m to $O\left(\frac{m}{\log(m)}\right)$, and that this is asymptotically tight. We also develop an $\tilde{O}(\sqrt[4]{m})$-competitive algorithm, providing evidence that known hardness results rely on "pathological" inputs, and complement it by proving that no algorithm can be $O\left(\frac{\log(m)}{\log\log(m)}\right)$-competitive in this model.

1.1 Related Results

The most classical Scheduling problem is *Makespan Minimization* on parallel and identical machines. Here, the goal is dual to Machine Covering; one wants to minimize the maximum load among the machines. This problem is strongly NP-hard and there exists a PTAS [27]. The online setting received considerable research attention, and already in 1966 Graham

showed that his famous Greedy-strategy is $(2 - 1/m)$-competitive. A long line of research [21, 11, 29, 1, 19] starting in the 1990s lead to the currently best competitive ratio of 1.9201 due to Fleischer and Wahl [19]. Regarding lower bounds, again after a sequence of results [17, 12, 24] the current best one is 1.88 [40].

The landscape for Online Machine Covering differs considerably from Online Makespan Minimization as discussed before, which has motivated the study of semi-online models to deal with the implied hard restrictions. If the value of the optimal minimum load of the instance is known in advance, Azar et al. have shown that a simple greedy algorithm already is $(2 - 1/m)$-competitive and that no algorithm can attain a competitive ratio better than 7/4 [7]. These bounds were improved by Ebenlendr et al. to 11/6 and 1.791 respectively [14]. In the *bounded migration* model, whenever a job of size p arrives, older jobs of total size at most $\beta \cdot p$ can be reassigned to different machines. Sanders et al. [41] provide a 2-competitive algorithm for $\beta = 1$. Later results [42, 22] study the interplay between improved competitive ratios and larger values of β. Another semi-online model provides the online algorithm with a *reordering buffer*, which is used to rearrange the input sequence "on the fly". Epstein et al. [16] provide a $(H_{m-1} + 1)$-competitive algorithm using a buffer of size $m - 1$, and show that this ratio cannot be improved for any sensible buffer size. These and many more semi-online models have also been studied for Makespan Minimization, see the survey in [15] and references therein.

The first Scheduling result in the random-order model is due to Osborn and Torng [39]. They establish that the Greedy-strategy for Makespan Minimization does not achieve a competitive ratio better than 2 for general m. Recently, [2, 3] show that for Makespan Minimization the random-order model allows for better performance guarantees than the classical model. Molinaro [38] has studied the Online Load Balancing problem with the objective to minimize general l_p-norms of the machine loads, providing an algorithm that returns solutions of expected norm $(1 + \varepsilon)\text{OPT} + O\left(\frac{p}{\varepsilon}(m^{1/p} - 1)\right)$ in the random-order model, where OPT denotes the optimal norm. Göbel et al. [23] have studied Average Weighted Completion Time Minimization on one machine in the random-order model. Their competitive ratio is logarithmic in the input length n for general job sizes and constant if all jobs have size 1. To the best of our knowledge, no previous result is known for Online Machine Covering in the random-order model.

1.2 Our Contribution

We first establish that the Greedy-strategy is $\Theta\left(\frac{m}{\log(m)}\right)$-competitive in the random-order model. This is only a tiny, albeit significant improvement compared to worst-case orders. Since the bound is tight, more refined strategies that make particular use of the characteristics of the random-order model are required. The analysis also gives first intuitions about what these characteristics are and also about the techniques used to analyze the main algorithm.

The following theorem summarizes the central result of this paper.

▶ **Theorem 1.** *There exists a $\tilde{O}(\sqrt[4]{m})$-competitive algorithm for the online Machine Covering problem in the random-order model.*

In the classical online Machine Covering problem, difficult instances are usually related to the inability of distinguishing "small" and "large" jobs induced by a lack of knowledge about large job classes hidden at the end of the sequence. Figure 1 depicts the easiest example on which deterministic schedulers cannot perform well as they cannot know that the first m jobs are tiny. Azar and Epstein [7] ameliorate this by maintaining a randomized threshold, which is used to distinguish small and large job sizes. They have to correctly classify up to m large

jobs with constant probability while controlling the total size of incorrectly classified small jobs, which leads to their randomized competitive ratio of $\tilde{O}(\sqrt{m})$. However, their lower bound shows that general randomized algorithms are still unable to schedule the first m jobs correctly with probability exceeding $\frac{1}{\sqrt{m}}$, again due to relevant job classes being hidden at the end of the input.

Random-order arrival makes such hiding impossible. This already helps the Greedy-strategy as now large jobs in the input are evenly distributed instead of being clustered at the end. Our $\tilde{O}(\sqrt[4]{m})$-competitive algorithm enhances the path described previously by making explicit use of the no-hiding-feature; it determines those large jobs the adversary would have liked to hide. Information about large job sizes is, as is common in Secretary problems, estimated in a sampling phase, which returns a threshold distinguishing all except for $\sqrt{m}$ of the large jobs. This reduction by a square root carries over to the competitive ratio: We now can allow to misclassify these remaining $\sqrt{m}$ jobs with a higher probability, which in turn leads to a better classification of small jobs and a better competitive ratio of $\tilde{O}(\sqrt[4]{m})$.

We complement the upper bound of $\tilde{O}(\sqrt[4]{m})$ with a lower bound of $\omega\left(\frac{\log(m)}{\log\log(m)}\right)$ for the competitive ratio in the random-order model. Lower bounds in the random-order model are usually considered hard to devise since one cannot hide larger pieces of input. Instead of hiding large job classes, we figuratively make them hard to distinguish by adding noise. To this end, we study a novel variant of the Secretary problem, the *Talent Contest* problem, where the goal is to find a good but not too good candidate (or secretary).

More in detail, we want to pick the K-th best among a randomly permuted input set of candidates. Unlike classical Secretary problems (or the more general Postdoc problem [43]), we may pick several candidates as long as they are not better than the K-th candidate. Furthermore, we interview candidates t times and make a decision at each arrival. In this setting, information gained by earlier interviews helps the decisions required in later ones. It can be proven that the expected number of times the desired candidate can be correctly identified relates to the ability of distinguishing exactly the $m-1$ largest jobs from a Machine Covering instance in the random-order model, and hence bounding the aforementioned expected value allows us to obtain the desired hardness result.

1.3   Organization of the paper

In Section 2 we provide the required definitions and tools, and then in Section 3 analyze the Greedy-algorithm in the context of random-order arrival. In Section 4 we present our main algorithm and its analysis. Section 5 then introduces the Talent Contest Problem and concludes with a lower bound for the best competitive ratio in the random-order model. Due to space constraints, some proofs from Section 5 are deferred to the full version of the paper.

2   Preliminaries

In this section we introduce the main definitions and tools that are used along this work.

In the Machine Covering problem, we are given n jobs $\mathcal{J} = \{J_1, \ldots, J_n\}$, specified by their non-negative sizes p_i, which are to be assigned onto m parallel and identical machines. The *load* l_M of a machine M is the sum of the sizes of all the jobs assigned to it. The goal is to maximize the minimum load among the machines, i.e. to maximize $\min_M l_M$.

To an online algorithm, jobs are revealed one-by-one and each has to be assigned permanently and irrevocably before the next one is revealed. Formally, given the symmetric group S_n on n elements, each permutation $\sigma \in S_n$ defines the order in which the elements of

$\mathcal{J}$ are revealed, namely $\mathcal{J}^\sigma = (J_{\sigma(1)}, \ldots, J_{\sigma(n)})$. Classically, the performance of an online algorithm A is measured in terms of competitive analysis. That is, if we denote the minimum machine load of A^1 on $\mathcal{J}^\sigma$ by $A(\mathcal{J}^\sigma)$ and the minimum machine load an optimal offline algorithm may achieve by $\mathrm{OPT}(\mathcal{J})$ (which is independent on the order σ), one is interested in finding a small *(adversarial) competitive ratio* $c = \sup_{\mathcal{J}} \sup_{\sigma \in S_n} \frac{\mathrm{OPT}(\mathcal{J})}{A(\mathcal{J}^\sigma)}$[2].

In the random-order model, the job order is chosen uniformly at random. We consider the permutation group S_n as a probability space under the uniform distribution. Then, given an input set $\mathcal{J}$ of size n, we charge A *random-order cost* $A^{\mathrm{rom}}(\mathcal{J}) = \mathbf{E}_{\sigma \sim S_n}[A(\mathcal{J}^\sigma)] = \frac{1}{n!} \sum_{\sigma \in S_n} A(\mathcal{J}^\sigma)$. The *competitive ratio in the random-order model* of A is $c = \sup_{\mathcal{J}} \frac{\mathrm{OPT}(\mathcal{J})}{A^{\mathrm{rom}}(\mathcal{J})}$. Throughout this work we will assume that n is known to the algorithm; this assumption is common in the literature and it can be proven that it does not help in the adversarial setting, see the full version of the article for details. When clear from the context, we will omit the dependency on $\mathcal{J}$.

Given $0 \leq i \leq n$, let $P_i = P_i[\mathcal{J}]$ refer to the size of the i-th largest job in $\mathcal{J}$. Given $i \geq 1$, we define $L_i := \sum_{j \geq i} P_j$ to be the total size of jobs smaller than P_i. Note that the terminology 'smaller' uses an implicit tie breaker since there may be jobs of equal sizes.

3 Properties and Analysis of the Greedy-strategy

We now proceed with some useful properties of Graham's Greedy-strategy. Recall that this algorithm always schedules an incoming job on some least loaded machine breaking ties arbitrarily. The following two lemmas recall useful standard properties of the algorithm, which will help us later to restrict ourselves to simpler special instances.

▶ **Lemma 2.** *The minimum load achieved by the Greedy-strategy is at least P_m.*

Proof. If the m largest jobs get assigned to different machines, the bound holds directly. On the other hand, if one of the m largest jobs J_j gets assigned to a machine that already had one of the m largest jobs $J_{j'}$, of size $P_{j'}$, then at the arrival of J_j the minimum load was at least $P_{j'} \geq P_m$, concluding the claim as the minimum load does not decrease through the iterations. ◀

▶ **Lemma 3.** *The minimum load achieved by the Greedy-strategy is, for each $i \in \{1, \ldots, m\}$, bounded below by $\frac{L_i}{m} - P_i$.*

Proof. For each machine M, let $J_{\mathrm{last}}(M)$ be the last job assigned to machine M. Let $\mathcal{J}_{\mathrm{last}}$ be the set of all these last jobs. If ALG denotes the minimum load achieved by the Greedy-strategy, then the load of each machine M in the schedule is at most $ALG + J_{\mathrm{last}}(M)$ as jobs are iteratively assigned to a least loaded machine. Now remove the $i - 1$ largest jobs in $\mathcal{J}_{\mathrm{last}}$ from the solution and let $\tilde{L}$ denote the total size of the remaining jobs in $\mathcal{J}_{\mathrm{last}}$. Then the total size of all remaining jobs is at most $m \cdot ALG + \tilde{L}$. On the other hand, since we only removed $i - 1$ jobs, the total size of the remaining jobs is at least L_i. Since $\tilde{L} \leq (m - i + 1) \cdot P_i$, we have that $L_i \leq m \cdot ALG + (m - i + 1) \cdot P_i$. Hence, the minimum load achieved by the Greedy-strategy is at least $\frac{L_i}{m} - \frac{m-i+1}{m} P_i \geq \frac{L_i}{m} - P_i$. ◀

With these tools we can now prove that the Greedy-strategy has a competitive ratio of at most $O\left(\frac{m}{H_m}\right)$ in the random-order model, where $H_m = \sum_{i=1}^m \frac{1}{i}$ denotes the m-th harmonic number. We also describe a family of instances showing that this analysis is asymptotically tight.

---

[1] In case A is randomized, we then refer to the expected minimum load.
[2] Using the convention that $0/0 = 1$ and $a/0 = \infty$ for $a > 0$.

Order $\mathcal{J}^\sigma$:

load $S_0[\mathcal{J}^\sigma]$ load $S_1[\mathcal{J}^\sigma]$ load $S_{\tilde{k}}[\mathcal{J}^\sigma]$
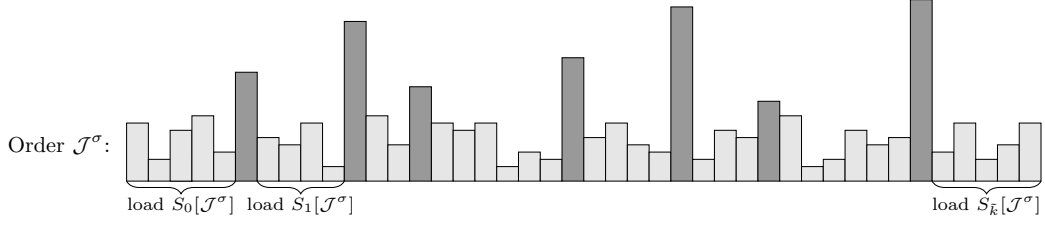
Figure 2 A possible order $\mathcal{J}^\sigma$ for an instance with $\tilde{k} < m$ large (dark) jobs. They partition the sequence into sets of small (light) jobs, each one having total size $S_i(\mathcal{J}^\sigma)$, $i = 0, \ldots, \tilde{k}$.

▶ **Theorem 4.** *The Greedy-strategy is* $(2 + o_m(1))\frac{m}{H_m}$*-competitive in the random-order model.*

Proof. Thanks to Lemma 2 we can assume that $P_m \leq \frac{H_m}{2m}\text{OPT}$. We say that a job is *large* if its size is larger than $\frac{H_m}{2m}\text{OPT}$, otherwise it is small. Let $\tilde{k} < m$ be the number of large jobs in the instance. Note that $L_{\tilde{k}+1} \geq (m - \tilde{k})\text{OPT}$ since in the optimal solution at most $\tilde{k}$ machines receive large jobs. Using Lemma 3 we may assume $\tilde{k} \geq m - H_m$, as otherwise we are done.

For a given order $\mathcal{J}^\sigma$ and $0 \leq i \leq \tilde{k}$, let $S_i(\mathcal{J}^\sigma)$ be the total size of small jobs preceded by precisely i large jobs with respect to $\mathcal{J}^\sigma$ (see Figure 2). We will prove that the minimum load achieved by the Greedy-strategy is at least $\min\left\{ \sum_{i=0}^{\tilde{k}} \frac{S_i(\mathcal{J}^\sigma)}{m-i} - \frac{H_m}{2m}\text{OPT}, \frac{H_m}{2m} \right\}\text{OPT}$.

A machine is said to be *full* once it receives a large job, and notice that we can assume that no full machine gets assigned further jobs as otherwise the minimum load would be already at least $\frac{H_m}{2m}\text{OPT}$. Consider the set of small jobs that arrive to the system before the first large job in the sequence. At this point the average load of the machines is exactly $\frac{S_0(\mathcal{J}^\sigma)}{m}$ and, since the upcoming large job gets assigned to a least loaded machine, the average load of the remaining machines is still at least $\frac{S_0(\mathcal{J}^\sigma)}{m}$. Now the upcoming small jobs that arrive before the second large job in the sequence get assigned only to these non-full machines, whose average load is now at least $\frac{S_0(\mathcal{J}^\sigma)}{m} + \frac{S_1(\mathcal{J}^\sigma)}{m-1}$. Since the following large job gets assigned to the least loaded of these machines, the average load of the remaining ones is still lower-bounded by this quantity. By iterating this argument, it can be seen that the average load of the machines containing only small jobs is at least $\frac{1}{m-\tilde{k}}\sum_{i=0}^{\tilde{k}} \frac{S_i(\mathcal{J}^\sigma)}{m-i}$. Since in a Greedy-strategy the load of two machines cannot differ by more than the size of the largest job assigned to them, we conclude that the minimum load achieved by the algorithm is at least $\sum_{i=0}^{\tilde{k}} \frac{S_i(\mathcal{J}^\sigma)}{m-i} - \frac{H_m}{2m}\text{OPT}$.

Now let us understand the random variable $S(\mathcal{J}^\sigma) = \sum_{i=0}^{\tilde{k}} \frac{S_i(\mathcal{J}^\sigma)}{m-i}$. If we pick $\sigma \sim S_n$ uniformly at random and consider the number $s(J)$ of large jobs that precede any fixed small job J, this number is uniformly distributed between 0 and $\tilde{k}$. Then we can rewrite $S(\mathcal{J}^\sigma) = \sum_{i=0}^{\tilde{k}} \frac{S_i(\mathcal{J}^\sigma)}{m-i} = \sum_{J \text{ small}} \frac{p_J}{m-s(J)}$. Since $\mathbf{E}[\frac{1}{m-s(J)}] = \sum_{i=0}^{\tilde{k}} \frac{1}{\tilde{k}+1}\frac{1}{m-i} = \frac{H_m - H_{m-\tilde{k}-1}}{\tilde{k}+1}$ we get by linearity of expectation that

$$
\begin{aligned}
\mathbf{E}[S(\mathcal{J}^\sigma)] &= \sum_{J \text{ small}} p_J \frac{H_m - H_{m-\tilde{k}-1}}{\tilde{k}+1} \\
&= \frac{H_m - H_{m-\tilde{k}-1}}{\tilde{k}+1} L_{\tilde{k}+1} \\
&\geq \frac{(m-\tilde{k})(H_m - H_{m-\tilde{k}-1})}{\tilde{k}+1} \frac{L_{\tilde{k}+1}}{m-\tilde{k}}.
\end{aligned}
$$

Notice that $\frac{L_{\tilde{k}+1}}{m-\tilde{k}}$ is a lower bound for OPT, which we will use later. The first factor is decreasing as a function of $\tilde{k} \leq m - 1$, and consequently the expression is at least $\frac{H_m}{m}$. Thus $\mathbf{E}[S(\mathcal{J}^\sigma)] \geq \frac{H_m}{m} \cdot \frac{L_{\tilde{k}+1}}{m-\tilde{k}}$.

We also get $\mathrm{Var}\left[\frac{1}{m-s(J)}\right] \leq \mathbf{E}\left[\left(\frac{1}{m-s(J)}\right)^2\right] = \sum_{i=0}^{\tilde{k}} \frac{1}{\tilde{k}+1} \frac{1}{(m-i)^2} \leq \frac{1}{\tilde{k}+1} \sum_i \frac{1}{i^2} \leq \frac{1}{\tilde{k}+1} \frac{\pi^2}{6}$.
Using that $\tilde{k} \geq m - H_m$, we broadly bound $\tilde{k}$ via $\tilde{k} + 1 \geq \frac{m}{2} \geq \frac{m(m-\tilde{k})^2}{2H_m^2}$. Substituting this
in the previous bound yields $\mathrm{Var}\left[\frac{1}{m-s(J)}\right] \leq \frac{H_m^2}{m} \frac{1}{(m-\tilde{k})^2} \frac{\pi^2}{3}$. Moreover, for two small jobs J_i
and J_j we have $\mathrm{Cov}\left[\frac{1}{m-s(J_i)}, \frac{1}{m-s(J_j)}\right] \leq \left(\mathrm{Var}\left[\frac{1}{m-s(J_i)}\right] \mathrm{Var}\left[\frac{1}{m-s(J_j)}\right]\right)^{1/2} \leq \frac{H_m^2}{m} \frac{1}{(m-\tilde{k})^2} \frac{\pi^2}{3}$.
For $i \neq j$ this bound is pessimistic. The correlation between $s(J_i)$ and $s(J_j)$ is positive but
tiny. We use this covariance to bound $\mathrm{Var}[S(\mathcal{J}^\sigma)] = \sum_{J \text{ small}} \frac{p_J}{m-s(J)}$.

$$
\begin{aligned}
\mathrm{Var}[S(\mathcal{J}^\sigma)] &= \sum_{J_i, J_j \text{ small}} p_i p_j \mathrm{Cov}\left[\frac{1}{m-s(J_i)}, \frac{1}{m-s(J_j)}\right] \\
&\leq \sum_{J_i \text{ small}} p_i \cdot \sum_{J_j \text{ small}} p_j \cdot \frac{H_m^2}{m} \frac{1}{(m-\tilde{k})^2} \frac{\pi^2}{3} \\
&\leq \frac{\pi^2}{3} \frac{H_m^2}{m} \left(\frac{L_{\tilde{k}+1}}{m-\tilde{k}}\right)^2.
\end{aligned}
$$

The last inequality uses again that $L_{\tilde{k}+1} \geq (m-\tilde{k})\mathrm{OPT}$. Hence, the standard deviation
$\mathrm{SD}[S(\mathcal{J}^\sigma)]$ is at most $C^{3/2} \frac{L_{\tilde{k}+1}}{m-\tilde{k}}$ for $C = \sqrt[3]{\frac{\pi^2}{3} \frac{H_m^2}{m}} = \Theta\left(\sqrt[3]{\frac{(\log(m))^2}{m}}\right)$. Chebyshev's inequality,
which allows to bound the probability of deviating from the mean in terms of the standard
deviation, yields

$$
\begin{aligned}
\mathbf{P}\left[S(\mathcal{J}^\sigma) \leq \left(\frac{H_m}{m} - C\right)\mathrm{OPT}\right] &\leq \mathbf{P}\left[S(\mathcal{J}^\sigma) \leq \left(\frac{H_m}{m} - C\right)\frac{L_{\tilde{k}+1}}{m-\tilde{k}}\right] \\
&\leq \mathbf{P}\left[|\mathbf{E}[S(\mathcal{J}^\sigma)] - S(\mathcal{J}^\sigma)| \geq C^{-1/2} \cdot \sigma[S(\mathcal{J}^\sigma)]\right] \\
&\leq C.
\end{aligned}
$$

We conclude that with probability $1 - C$ the minimum load achieved by the Greedy-
strategy exceeds $\min\left\{S(\mathcal{J}^\sigma) - \frac{H_m}{2m}, \frac{H_m}{2m}\right\}\mathrm{OPT} \geq \frac{H_m}{m}\left(\frac{1}{2} - C\right)\mathrm{OPT}$. Thus its competitive
ratio in the random-order model is at most $(1-C)\frac{m}{H_m}\frac{1}{(\frac{1}{2}-C)} = (2 + o_m(1))\frac{m}{H_m}$. ◀

▶ **Theorem 5.** *The Greedy-strategy is not better than $\frac{m}{H_m}$-competitive in the random-order
model.*

Proof. Consider the following instance for $\varepsilon > 0$: Job set $\mathcal{J}$ consists of $m - 1$ jobs of
size 1 and $\frac{1}{\varepsilon}$ jobs of size ε. It is not difficult to see that $\mathrm{OPT} = 1$ by assigning the jobs
of size 1 to different machines and the jobs of size ε together on the remaining machine.
Following a similar approach as the one above, interpreting the jobs of size 1 as large and
the rest as small, we can prove that for any given order $\mathcal{J}^\sigma$, the expected minimum load
achieved by the Greedy-strategy is at most $\sum_{i=0}^{m-1} \frac{S_i(\mathcal{J}^\sigma)}{m-i} + \varepsilon m$. If now $\sigma \sim S_n$ is chosen
uniformly at random, then the minimum load achieved by the Greedy-strategy is at most
$\sum_{i=0}^{m-1} \frac{1}{m(m-i)} + \varepsilon m = \frac{H_m}{m} + \varepsilon m$. In particular, the competitive ratio in the random-order
model of the Greedy-strategy is at most $1/(\frac{H_m}{m} + \varepsilon m)$ which approaches $\frac{m}{H_m}$ as $\varepsilon \to 0$. ◀

4 An $\tilde{O}(\sqrt[4]{m})$-competitive algorithm

We now describe an improved competitive algorithm for the problem further exploiting the
extra features of the random-order model. More in detail, we devise a sampling-based method
to classify relatively large jobs (with respect to OPT). After this, we run a slight adaptation
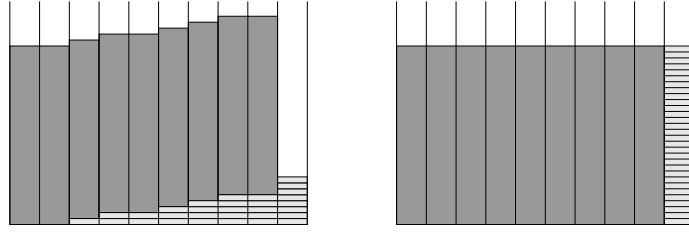
Figure 3 A comparison of the solution returned by the Greedy-strategy (left) and the optimal solution (right) for some order σ of the instance defined in Theorem 4. Dark jobs have size 1 and the remaining jobs have size $\varepsilon > 0$.

of the algorithm *Partition* due to Azar and Epstein [7] in order to distinguish large jobs that our initial procedure could not classify, leading to strictly better approximation guarantees (see Algorithm 1 for a description). We will assume w.l.o.g. that job sizes are rounded down to powers of 2, which induces an extra multiplicative factor of at most 2 in the competitive ratio.

4.1   Simple and Proper Inputs

Given an input sequence $\mathcal{J}$, we call any job J *large* if its size is larger than $\frac{\mathrm{OPT}[\mathcal{J}]}{100 \sqrt[4]{m}}$, and we call it *small* otherwise. Let $k = k[\mathcal{J}]$ be the number of large jobs in $\mathcal{J}$. Let $L_{\mathrm{small}} = L_{k+1}$ be the total size of small jobs. The following definition allows to recognize instances where the Greedy-strategy performs well, see Proposition 7.

▶ **Definition 6.** *We call the input set $\mathcal{J}$ **simple** if either*
- *The set $\mathcal{J}$ has size $n < m$,*
- *There are at least m large jobs, i.e. $k \geq m$, or*
- *There are at most $m - \frac{\sqrt[4]{m^3}}{50}$ large jobs, i.e. $k \leq m - \frac{\sqrt[4]{m^3}}{50}$.*

Note that the first condition is mostly included for ease of notation; the third condition implies that sequences with $n < m - \frac{\sqrt[4]{m^3}}{50}$ are simple, which is good enough for our purposes but somewhat clumsy to refer to.

▶ **Proposition 7.** *The Greedy-strategy achieves minimum load at least $\frac{\mathrm{OPT}}{100 \sqrt[4]{m}}$ on simple inputs.*

Proof. We consider each case from Definition 6 separately:
- If the instance has less than m jobs, $\mathrm{OPT} = 0$ and every algorithm is optimal.
- If there are at least m large jobs in the instance, then the minimum load achieved by the Greedy algorithm is at least $P_m \geq \frac{\mathrm{OPT}}{100 \sqrt[4]{m}}$ thanks to Lemma 2.
- If there are at most $m - \frac{\sqrt[4]{m^3}}{50}$ large jobs, then $L_{k+1} \geq \frac{\sqrt[4]{m^3}}{50}\mathrm{OPT}$ as there are at least $\frac{\sqrt[4]{m^3}}{50}$ machines without large jobs in the optimal solution. If we apply Lemma 3 with $i = k + 1$, the minimum load achieved by the Greedy algorithm is at least $\frac{\mathrm{OPT}}{100 \sqrt[4]{m}}$.   ◄

For an input set $\mathcal{J}$ which is not simple, let $d := \lceil \log_2(m - k) \rceil$. Note that $0 \leq d \leq \left\lceil \frac{3}{4}\log(m) \right\rceil$. We say that such an instance $\mathcal{J}$ is *proper (of degree d)*.

Algorithm 1 guesses a value t with probability $\Omega(1/\log(m))$ to address in a different way simple instances (case $t = -1$) and proper instances of degree t for each $0 \leq t \leq \left\lceil \frac{3}{4}\log(m) \right\rceil$, at the expense of an extra logarithmic factor in the competitive ratio. By Proposition 7, simple instances are sufficiently handled by the Greedy-strategy. From now on we will focus on proper instances of degree d and show that the corresponding case when $t = d$ in Algorithm 1 returns a $O\left(\sqrt[4]{m} \right)$-approximate solution.

Algorithm 1 The online Algorithm for Random-Order Machine Covering.

Input: Job sequence $\mathcal{J}^\sigma$, m identical parallel machines.

1: Guess $t \in \{-1, 0, 1, \ldots, \lceil \frac{3}{4} \log(m) \rceil\}$ uniformly at random.
2: **if** $t = -1$, **then** Run the Greedy-strategy and **return** the computed solution.
3: **else** Partition the machines into 2^t *small* machines and $m - 2^t$ *large* machines.

Phase 1 – Sampling.

4: Schedule the first $n/8$ jobs iteratively into a least loaded large machine.
5: Let $P^\uparrow$ be the $\left(\frac{m-2^t}{8} - \frac{\sqrt{m}}{2} \right)$-th largest job size among these first $n/8$ jobs.

Phase 2 – Partition.

6: $\tau \leftarrow 0$
7: **for** $j = \frac{n}{8} + 1, \ldots, n$ **do**
8:     **if** $p_{\sigma(j)} \geq P^\uparrow$, **then** Schedule job $J_{\sigma(j)}$ onto a least loaded large machine.
9:     **if** $p_{\sigma(j)} > \tau$, **then** Update τ to $p_{\sigma(j)}$ with probability $\frac{1}{9 \cdot 2^t \sqrt{m}}$.
10:     **if** $p_{\sigma(j)} \leq \tau$, **then** Schedule job $J_{\sigma(j)}$ onto a least loaded small machine.
11:     **if** $p_{\sigma(j)} > \tau$, **then** Schedule job $J_{\sigma(j)}$ onto a least loaded large machine.
12: **end for**
13: **return** the computed solution.

## 4.2 Algorithm for Proper Inputs of Degree d

Let $\mathcal{J}$ be proper of degree d and assume without loss of generality that its size n is divisible by 8 (an online algorithm can always simulate up to 7 extra jobs of size 0 to reduce to this case). The algorithm, assuming d is guessed correctly, chooses 2^d *small* machines $\mathcal{M}_{\text{small}}$, while the other machines $\mathcal{M}_{\text{large}}$ are called *large machines*. The algorithm will always assign the incoming job either to a least loaded small or to a least loaded large machine, the only choice it has to make is to which set of machines the job will go. Theoretically, the goal would be to assign all large jobs to large machines and all small ones to small machines according to our definition. Large and small jobs, unfortunately, cannot be distinguished by an online algorithm with certainty. Instead, we have to use randomization and expect a small error. We aim for a small one-sided error, meaning that we want to avoid misclassifying large jobs at all cost while incorrectly labeling very few small jobs as large.

The algorithm starts with a *sampling phase*: the first $\frac{n}{8}$ jobs will be used for sampling purposes, and since we yet lack good knowledge about what should be considered large, these jobs will all be assigned to large machines. Let $P^\uparrow$ be the element of rank $\frac{m-2^d}{8} - \frac{\sqrt{m}}{2}$ among these elements. The following lemma shows that $P^\uparrow$ can be used as a threshold to distinguish most of the large jobs from small ones.

▶ **Lemma 8.** *For proper sequences, it holds that* $\mathbf{P}[P_{k-8\sqrt{m}-2^d} \geq P^\uparrow \geq P_k] \geq \frac{1}{3}$.

Proof. Let n_{large} be the number of large jobs among the first $\frac{n}{8}$ jobs in the sequence. Notice that n_{large} obeys an hypergeometric distribution with parameters $N = n$ (size of the total population), $K = k$ (number of elements with the desired property) and $r = \frac{n}{8}$ (size of the sample). This implies that $\mathbf{E}_{\sigma \sim S_n}[n_{\text{large}}] = \frac{k}{8} \geq \frac{m-2^d}{8}$. Moreover

$$\text{Var}[n_{\text{large}}] = \frac{n}{8} \cdot \frac{k}{n} \cdot \frac{n-k}{n} \cdot \frac{7n}{8(n-1)} = \frac{7}{64} \cdot \frac{k(n-k)}{n-1} < \frac{m}{8}.$$
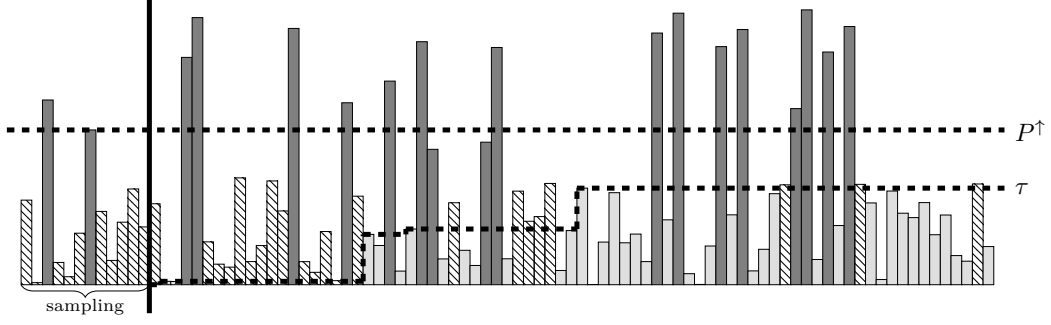
Figure 4 The classification of large (dark) and small (light and dashed) jobs. During sampling, all small jobs are misclassified (dashed ones). Threshold $P^\uparrow$ classifies large jobs, while threshold τ classifies small jobs. Jobs in between are conservatively classified as large, since misclassifying large jobs is fatal. Increasing τ due to a small job is a helpful event as less small jobs will be misclassified. On the other hand, increasing τ due to a large job below $P^\uparrow$ is a fatal event as large jobs will be misclassified. The choice of $P^\uparrow$ ensures that fatal events are unlikely to happen.

If we use Cantelli's inequality, a one-sided version of Chebyshev's inequality, then

$$\mathbf{P}[P^\uparrow < P_k] = \mathbf{P}\left[n_{\text{large}} < \frac{m - 2^d}{8} - \frac{\sqrt{m}}{2}\right] \leq \mathbf{P}\left[n_{\text{large}} < \mathbf{E}[n_{\text{large}}] - \frac{\sqrt{m}}{2}\right] \leq \frac{m/8}{\frac{m}{8} + \frac{m}{4}} = \frac{1}{3}.$$

Consider now n'_{large} to be the number of the $k - 8\sqrt{m} - 2^d$ largest jobs among the $\frac{n}{8}$ first jobs in the sequence. Similarly as before, n'_{large} obeys an hypergeometric distribution with parameters $N = n$, $K = k - 8\sqrt{m} - 2^d$ and $r = \frac{n}{8}$, which implies that $\mathbf{E}_{\sigma \sim S_n}[n'_{\text{large}}] = \frac{k - 2^d - 8\sqrt{m}}{8} < \frac{m - 2^d}{8} - \sqrt{m}$. Here we use that $k \leq m$. Furthermore $\text{Var}[n'_{\text{large}}] = \frac{n}{8} \cdot \frac{k - 8\sqrt{m} - 2^d}{n} \cdot \frac{n - k + 8\sqrt{m} + 2^d}{n} \cdot \frac{7n}{8(n-1)} \leq \frac{m}{8}$. Then, using again Cantelli's inequality, we obtain that

$$\mathbf{P}\left[P_{k - 8\sqrt{m} - 2^d} < P^\uparrow\right] = \mathbf{P}\left[n'_{\text{large}} \geq \frac{m - 2^d}{8} - \frac{\sqrt{m}}{2}\right]$$
$$\leq \mathbf{P}\left[n'_{\text{large}} \geq \mathbf{E}[n'_{\text{large}}] + \frac{\sqrt{m}}{2}\right] \leq \frac{m/8}{\frac{m}{8} + \frac{m}{4}} = \frac{1}{3}.$$

We conclude that $\mathbf{P}[P_{k - 8\sqrt{m} - 2^d} \geq P^\uparrow \geq P_k] = 1 - \mathbf{P}\left[P_{k - 8\sqrt{m} - 2^d} < P^\uparrow\right] - \mathbf{P}[P^\uparrow < P_k] \geq 1/3$. ◀

After the aforementioned sampling phase is finished and $P^\uparrow$ is known, the algorithm will enter a *partition phase*. If a job now has size at least $P^\uparrow$, it is assigned to the large machines as it will be large with high probability. The large jobs below this value are the most difficult to assign. To this end, we define a *threshold value* τ, which we initialize to 0. If the incoming job has size at most τ we simply assign it to a least loaded small machine, but whenever we encounter a job J of size $p > \tau$, we set $\tau = p$ with probability $\frac{1}{9 \cdot 2^d \sqrt{m}}$. If now $p \leq \tau$, which could happen if we just increased τ, we schedule J on the least loaded small machine. Else, J is assigned to the least loaded machine in $\mathcal{M}_{\text{large}}$ (see Figure 4 for a depiction of the procedure). As the following lemma shows, this procedure distinguishes all the large jobs with constant probability.

▶ **Lemma 9.** *All large jobs are scheduled onto large machines with constant probability.*

Proof. Let us assume that $P_{k-8\sqrt{m}-2^d} \geq P^\uparrow \geq P_k$. Now, the statement of the lemma can only be wrong if we decided to increase τ when encountering some large job of size less than $P^\uparrow$. By assumption there are less than $8\sqrt{m} + 2^d$ such jobs. The desired probability is thus at least

$$\left(1 - \frac{1}{9 \cdot 2^d \sqrt{m}}\right)^{8\sqrt{m}+2^d} \geq 1 - \frac{8\sqrt{m}+2^d}{9 \cdot 2^d \sqrt{m}} \geq 1 - \frac{8}{9 \cdot 2} - \frac{1}{9\sqrt{m}} > \frac{4}{9}.$$

The first inequality is Bernoulli's inequality, the second one uses that $d \geq 1$. The lemma follows by multiplying with the probability from Lemma 8. ◄

We call an input sequence *orderly* if it satisfies the properties of Lemmas 8 and 9. The following lemma shows that the total size of misclassified small jobs can be, in expectation, bounded from above. This proof is an adaptation of one of the results from Azar and Epstein [7], which we present for the sake of completeness.

▶ **Lemma 10.** *The expected total size of small jobs scheduled onto small machines is at least $\frac{31}{800\sqrt[4]{m}}L_{\text{small}}$, even when conditioned on the input sequence being orderly.*

Proof. Let L'_{small} be the random variable corresponding to the size of small jobs assigned to large machines in the sampling phase. Since jobs appear in random order, we have that

$$\mathbf{E}_{\sigma \sim S_n}[L'_{\text{small}}] = \frac{1}{n!} \sum_{J_i \text{ small}} \frac{n}{8} \cdot (n-1)! \cdot p_i = \frac{1}{8}L_{\text{small}}.$$

Let us now bound the total size of misclassified small jobs in the partition phase. We will define, for a given set of $18 \cdot 2^d \sqrt[4]{m}$ small jobs of the same size p_i (recall that jobs are rounded down to powers of 2), the following event: after $9 \cdot 2^d \sqrt[4]{m}$ jobs from the set arrived, τ is at least p_i. If this were not the case, τ was never updated at any of these $9 \cdot 2^d \sqrt[4]{m}$ first jobs albeit being smaller than p_i. The probability for this is at most

$$\left(1 - \frac{1}{9 \cdot 2^d \sqrt{m}}\right)^{9 \cdot 2^d \sqrt[4]{m}} \leq e^{-\frac{1}{\sqrt[4]{m}}} \leq 1 - \frac{1}{2\sqrt[4]{m}},$$

where we used the fact that $e^{-x} \leq 1 - \frac{x}{2}$ if $0 \leq x \leq 1$. This implies that the probability of the previous event occurring is at least $\frac{1}{2\sqrt[4]{m}}$.

Let $\mathcal{S}$ be the set of small jobs remaining after the sampling phase. We will partition $\mathcal{S}$ into batches of $18\sqrt[4]{m}$ jobs of the same size. There will be jobs that are not assigned to any batch because there are not enough jobs of the same size to complete it, but the total size of these jobs is at most

$$18 \cdot 2^d \sqrt[4]{m} \sum_{i \geq 1} \frac{\text{OPT}}{100\sqrt[4]{m} \cdot 2^i} \leq \frac{18 \cdot 2^{d-1} \cdot \text{OPT}}{25} \leq \frac{18L_{\text{small}}}{25},$$

where the last inequality holds as there are at least 2^{d-1} machines in the optimal solution without large jobs. For each of the batches, the probability of assigning at least half of its size to small machines is bounded below by the probability of the previously described event occurring, which is at least $\frac{1}{2\sqrt[4]{m}}$. Hence, the expected total size of small jobs assigned to small machines is at least

$$\frac{1}{2\sqrt[4]{m}} \cdot \frac{1}{2} \sum_{J_i \in \mathcal{S}} p_i \geq \frac{1}{4\sqrt[4]{m}} \cdot \left(1 - \frac{1}{8} - \frac{18}{25}\right) L_{\text{small}} = \frac{31}{800\sqrt[4]{m}}L_{\text{small}}.$$

To observe that we can condition on the sequence being orderly, it suffices to note that the arguments work for every way to fix $P^\uparrow$ and that they do not make any assumptions on τ being increased at large jobs. ◄

Putting all the previous ingredients together we can conclude the following proposition.

▶ **Proposition 11.** *The previously described algorithm is $O\left(\sqrt[4]{m}\right)$-competitive in the random-order model for the case of proper inputs of degree d.*

Proof. By assumption, all $k \geq m - 2^d$ large jobs are scheduled onto large machines. A lower bound of $P_k = \frac{\mathrm{OPT}}{100\sqrt[4]{m}}$ for the minimum load achieved in the large machines follows then from Lemma 2. By Lemma 3, the minimum load among small machines is at least $\frac{31}{800\sqrt[4]{m}}\frac{L_{\mathrm{small}}}{2^d} - \frac{\mathrm{OPT}}{100\sqrt[4]{m}}$. The proposition follows from observing that $L_{\mathrm{small}} \geq 2^{d-1}\mathrm{OPT}$ since the optimal solution contains at least $m - k \geq 2^{d-1}$ machines with only small jobs. ◀

4.3 The Final Algorithm

As discussed before, our final algorithm first guesses whether the instance is simple or proper of degree d. Then we apply the appropriate algorithm, the Greedy-strategy or the previously described algorithm for the right degree. Since there are $O(\log(m))$ many possibilities, this guessing induces an extra logarithmic factor on the final competitive ratio, which concludes the proof of Theorem 1 restated below.

▶ **Theorem 1.** *There exists a $\tilde{O}(\sqrt[4]{m})$-competitive algorithm for the online Machine Covering problem in the random-order model.*

5 A Lower Bound for the Random-Order Model

The main difficulty for Online Machine Covering algorithms, including our main result, is to tell large jobs apart from the largest small jobs. In this section we prove that doing so is, to a certain extent, inherently hard. The main difference to adversarial models is that hardness is not obtained through withholding information but rather through obscuring it. This relates to some studied variants of the classical Secretary Problem such as the Postdoc Problem [43] but requires additional features particularly catered to our needs.

5.1 The Talent Contest Problem

Consider the following selection problem: To a yearly talent show contest n candidates apply. To appeal to a general audience, we try to exclude the best candidates because an imperfect performance is more entertaining, but we also want to have at least an appropriate candidate who can be presented as the winner. To do so, each candidate will participate in t trials (each trial is considered as an arrival) and we must decide for every arrival if we mark the candidate or not, meaning that we consider her to be the K-th best candidate or worse. The global order in which candidates arrive for trials is uniformly distributed, thus at later trials we have much more information to go by. Our final goal is to maximize the number of trials for which we successfully marked the K-th best candidate without marking any better candidate.

Formally, the *Talent Contest* problem is specified by three parameters K, n and t, where $K \leq n$. Candidates have pairwise different non-negative valuations $v_1, v_2, \ldots, v_n$, and each candidate arrives t times; the arrival order is chosen uniformly at random. The valuation of each candidate is revealed when the candidate arrives for the first time. We may decide to mark each arrival or not, though once the next candidate arrives such marking decision is permanent. For each value $1 \leq h \leq t$ we get one point if we marked the h-th arrival of the K-th best candidate, but not the h-th arrival of any better candidate. In

particular, we can get up to t points in total, one for each value of h. Let $P(K, t, n)$ be the expected number of points the optimal online algorithm scores given the three values K, t and n. Similar to the classical Secretary problem, we are mostly interested in the limit case $P(K, t) = \lim_{n \to \infty} P(K, t, n)$.

We require for our desired results one extra technical definition. Given $\lambda \geq 1$, we call the valuations of candidates λ-steep if all candidate valuations are guaranteed to be at least by a factor λ apart, i.e. $\min_{v_i > v_j} \frac{v_i}{v_j} \geq \lambda$. It is possible to prove the following bound on the expected value $P(K, t)$, whose proof we defer to the full version of the paper.

▶ **Lemma 12.** *It holds that* $P(K, t) \leq \frac{\zeta(t/2)(t+1)^{t/2}}{2\pi\sqrt{K}}$, *where ζ is the Riemann Zeta Function. This bound still holds if we restrict ourselves to λ-steep valuations for some $\lambda \geq 1$.*

Roughly speaking, the proof relies on the fact that if an algorithm manages to perform relatively well in the Talent Contest problem, then it could be used to guess the value of a binomially distributed random variable. For the latter problem, difficulty can be directly established. However, the proof involves more technical aspects as some few irregular orders do not allow this reduction and have to be excluded beforehand. The property of λ-steepness is ensured by choosing μ sufficiently large and applying the transformation $v \mapsto \mu^v$ to each valuation.

5.2 Reduction to Machine Covering

It is possible to show that the Talent Contest problem and the Online Machine Covering problem in the random-order model are related as the following lemma states.

▶ **Lemma 13.** *Given K and t, let $m = (K - 1) \cdot t + 1$. No (possibly randomized) algorithm for Machine Covering in the random-order model on m machines can be better than $\frac{t}{P(K,t)+1}$-competitive.*

Proof. Let $\lambda > t$. Consider any instance of the Talent Contest problem with λ-steep valuations. We will treat the arrival sequence of candidates as a job sequence, where each arrival corresponds to a job of size given by the valuation of the corresponding candidate. We call the $m - 1 = t(K - 1)$ jobs corresponding to the arrivals of the K largest candidates *large*. The t jobs corresponding to the next candidate are called *medium*. Notice that the size of a medium job is at most $\frac{OPT}{t}$ as evidenced by the schedule that assigns each large job on a separate machine and the t medium jobs onto the single remaining machine. Jobs which are neither large nor medium have total size at most $t \sum_{i=1}^{\infty} \lambda^{-i} \frac{\text{OPT}}{t} = \frac{\text{OPT}}{\lambda - 1}$ and are thus called *small*. They will become negligible for $\lambda \to \infty$.

Consider an online algorithm $\mathcal{A}_{MC}$ for Machine Covering in the random-order model. We will derive an algorithm $\mathcal{A}_{TC}$ for the Talent Contest problem as follows: $\mathcal{A}_{TC}$ marks each job that gets assigned to a machine that already contains a job of the same size. Let P be the number of points this strategy gets. We will first show that the schedule of $\mathcal{A}_{MC}$ contains a machine which has at most $P + 1$ medium jobs and no big job. For this we consider any fixed input order, and if $\mathcal{A}_{MC}$ is randomized, we consider any fixed outcome of its coin tosses.

For $2 \leq i \leq t$, let w_i be an indicator variable that is 1 if $\mathcal{A}_{TC}$ gains a point for the i-th arrival, i.e. if it marks the i-th arrival of the K-th best candidate but not the i-th arrival of a better candidate; $w_i = 0$ otherwise. Let also r_i be an indicator variable that is 1 if $\mathcal{A}_{TC}$ marked the i-th arrival of the K-th best candidate but still loses due to also marking the i-th arrival of a better candidate. Finally, let $\mathcal{M}_{\text{small}}$ be the machines which do not receive a large job in the schedule of $\mathcal{A}_{MC}$, and let Z_{med} be the average number of medium jobs on machines in $\mathcal{M}_{\text{small}}$. Our intermediate goal is to show that $Z_{\text{med}} < 2 + \sum_{i=2}^{t} w_i$.

Since there are only $m-1$ large jobs, of which at least $\sum_{i=2}^{t} r_i$ are scheduled on a machine already containing a large job, we have that $|\mathcal{M}_{\text{small}}| \geq 1 + \sum_{i=2}^{t} r_i$. Let $d \geq 0$ such that $|\mathcal{M}_{\text{small}}| = 1 + d + \sum_{i=2}^{t} r_i$. Now, observe that $\sum_{i=2}^{t} (w_i + r_i)$ counts the number of medium jobs that are placed on a machine already containing a medium job. Thus, the number of medium jobs on machines in $\mathcal{M}_{\text{small}}$ is at most $|\mathcal{M}_{\text{small}}| + \sum_{i=2}^{t} (w_i + r_i) = 1 + d + \sum_{i=2}^{t} (w_i + 2r_i)$. Now we can bound the average number of medium jobs on $\mathcal{M}_{\text{small}}$, namely $Z_{\text{med}} \leq \frac{1+d+\sum_{i=2}^{t}(w_i+2r_i)}{1+d+\sum_{i=2}^{t} r_i}$. Let us assume that $Z_{\text{med}} \geq 2$; then the term on the right hand side increases if we set d and all r_i to zero, and we obtain that $Z_{\text{med}} \leq 1 + \sum_{i=2}^{t} w_i$ and thus $Z_{\text{med}} < 2 + \sum_{i=2}^{t} w_i$. To derive this inequality we assumed that $Z_{\text{med}} \geq 2$ but it is trivially true if $Z_{\text{med}} < 2$.

Now, let z_{med} be the least number of medium jobs on a machine in $\mathcal{M}_{\text{small}}$. Then $z_{\text{med}} \leq Z_{\text{med}} < 2 + \sum_{i=2}^{t} w_i$. Since z_{med} and the right hand side are both integers, it holds that $z_{\text{med}} \leq 1 + \sum_{i=2}^{t} w_i$. Since $\sum_{i=2}^{t} w_i = P$, the number of points obtained by algorithm $\mathcal{A}_{TC}$ for the Talent Contest problem, we have shown that the schedule of $\mathcal{A}_{MC}$ contains a machine M with at most $P+1$ medium jobs and no large one as desired.

As argued before, each medium job has size at most $\frac{\text{OPT}}{t}$ and the small jobs have total size at most $\frac{\text{OPT}}{\lambda-1}$ in total. Thus, machine M has load at most $\frac{P+1}{t}\text{OPT} + \frac{\text{OPT}}{\lambda-1}$. In conclusion, the expected load of the least loaded machine in the schedule of $\mathcal{A}_{MC}$ is at most $\frac{P(K,t)+1}{K}\text{OPT} + \frac{\text{OPT}}{\lambda-1}$, given a worst-case input for the Talent Contest Problem. This concludes the proof by taking $\lambda \to \infty$.   ◀

By setting $K = (t+1)^t$ and combining this with the lower bound in Lemma 12, we obtain the following general lower bound.

▶ **Theorem 14.** *The competitive ratio of no online algorithm for Machine Covering in the random-order model, deterministic or randomized, is better than $\frac{\lfloor e^{W(\ln(m))}\rfloor - 1}{1.16+o(1)}$. Here, $W(x)$ is the Lambert W-function, i.e. the inverse to $x \mapsto xe^x$. In particular, no algorithm can be $O\left(\frac{\log(m)}{\log\log(m)}\right)$-competitive for Online Machine Covering in the random-order model.*

Proof. Let $K = (t+1)^t$. Then Lemma 12 yields $P(K,t) \leq \frac{\zeta(t/2)}{2\pi} \leq 1.16+o(1)$. By Lemma 13 no algorithm can be better than $\left(\frac{t}{1.16+o(1)}\right)$-competitive for $m = (K-1) \cdot t + 1 < (t+1)^{t+1}$. We can always add a few jobs of large enough size so that the lower bound extends to larger numbers of machines. The theorem follows since the inverse function of $x \mapsto (t+1)^{t+1}$ is $t \mapsto e^{W(\ln(m))} - 1$; the second part uses the identity $W(x) \geq \log(x) - \log\log(x) + \omega(1)$, see [28].   ◀

---- **References** ----

1 Susanne Albers. Better bounds for online scheduling. *SIAM J. Comput.*, 29(2):459–473, 1999. `doi:10.1137/S0097539797324874`.

2 Susanne Albers and Maximilian Janke. Scheduling in the random-order model. In *47th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 168, pages 68:1–68:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. `doi:10.4230/LIPIcs.ICALP.2020.68`.

3 Susanne Albers and Maximilian Janke. Scheduling in the secretary model. In *41st Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, 2021. To appear.

4 Susanne Albers, Arindam Khan, and Leon Ladewig. Best fit bin packing with random order revisited. In *45th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 170, pages 7:1–7:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. `doi:10.4230/LIPIcs.MFCS.2020.7`.

5 Susanne Albers, Arindam Khan, and Leon Ladewig. Improved online algorithms for knapsack and GAP in the random order model. *Algorithmica*, 83(6):1750–1785, 2021. `doi:10.1007/s00453-021-00801-2`.

6 Arash Asadpour and Amin Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. *SIAM J. Comput.*, 39(7):2970–2989, 2010. `doi:10.1137/080723491`.

7 Yossi Azar and Leah Epstein. On-line machine covering. *Journal of Scheduling*, 1(2):67–77, 1998. `doi:10.1002/(SICI)1099-1425(199808)1:2<67::AID-JOS6>3.0.CO;2-Y`.

8 Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. A knapsack secretary problem with applications. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 10th International Workshop (APPROX/RANDOM)*, volume 4627, pages 16–28. Springer, 2007. `doi:10.1007/978-3-540-74208-1_2`.

9 Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. Matroid secretary problems. *J. ACM*, 65(6):35:1–35:26, 2018. `doi:10.1145/3212512`.

10 Nikhil Bansal and Maxim Sviridenko. The santa claus problem. In *38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 31–40. ACM, 2006. `doi:10.1145/1132516.1132522`.

11 Yair Bartal, Amos Fiat, Howard J. Karloff, and Rakesh Vohra. New algorithms for an ancient scheduling problem. *J. Comput. Syst. Sci.*, 51(3):359–366, 1995. `doi:10.1006/jcss.1995.1074`.

12 Yair Bartal, Howard J. Karloff, and Yuval Rabani. A better lower bound for on-line scheduling. *Inf. Process. Lett.*, 50(3):113–116, 1994. `doi:10.1016/0020-0190(94)00026-3`.

13 Eugene B. Dynkin. The optimum choice of the instant for stopping a Markov process. *Soviet Math. Dokl*, 4, 1962.

14 Tomás Ebenlendr, John Noga, Jirí Sgall, and Gerhard J. Woeginger. A note on semi-online machine covering. In *Approximation and Online Algorithms, Third International Workshop (WAOA)*, volume 3879, pages 110–118. Springer, 2005. `doi:10.1007/11671411_9`.

15 Leah Epstein. A survey on makespan minimization in semi-online environments. *J. Sched.*, 21(3):269–284, 2018. `doi:10.1007/s10951-018-0567-z`.

16 Leah Epstein, Asaf Levin, and Rob van Stee. Max-min online allocations with a reordering buffer. *SIAM J. Discret. Math.*, 25(3):1230–1250, 2011. `doi:10.1137/100794006`.

17 Ulrich Faigle, Walter Kern, and György Turán. On the performance of on-line algorithms for partition problems. *Acta Cybern.*, 9(2):107–119, 1989.

18 Moran Feldman, Ola Svensson, and Rico Zenklusen. A simple $O(\log \log(\mathrm{rank}))$-competitive algorithm for the matroid secretary problem. *Math. Oper. Res.*, 43(2):638–650, 2018. `doi:10.1287/moor.2017.0876`.

19 Rudolf Fleischer and Michaela Wahl. Online scheduling revisited. In *Algorithms, 8th Annual European Symposium (ESA)*, volume 1879, pages 202–210. Springer, 2000. `doi:10.1007/3-540-45253-2_19`.

20 Donald K. Friesen and Bryan L. Deuermeyer. Analysis of greedy solutions for a replacement part sequencing problem. *Math. Oper. Res.*, 6(1):74–87, 1981. `doi:10.1287/moor.6.1.74`.

21 Gábor Galambos and Gerhard J. Woeginger. An on-line scheduling heuristic with better worst case ratio than graham's list scheduling. *SIAM J. Comput.*, 22(2):349–355, 1993. `doi:10.1137/0222026`.

22 Waldo Gálvez, José A. Soto, and José Verschae. Symmetry exploitation for online machine covering with bounded migration. *ACM Trans. Algorithms*, 16(4):43:1–43:22, 2020. `doi:10.1145/3397535`.

23 Oliver Göbel, Thomas Kesselheim, and Andreas Tönnis. Online appointment scheduling in the random order model. In *Algorithms, 23rd Annual European Symposium (ESA)*, volume 9294, pages 680–692. Springer, 2015. `doi:10.1007/978-3-662-48350-3_57`.

24 Todd Gormley, Nick Reingold, Eric Torng, and Jeffery R. Westbrook. Generating adversaries for request-answer games. In *11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 564–565. ACM/SIAM, 2000.

25 Anupam Gupta, Ruta Mehta, and Marco Molinaro. Maximizing profit with convex costs in the random-order model. In *45th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 107, pages 71:1–71:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. `doi:10.4230/LIPIcs.ICALP.2018.71`.

26 Anupam Gupta and Sahil Singla. Random-order models. In *Beyond the Worst-Case Analysis of Algorithms*, pages 234–258. Cambridge University Press, 2020. `doi:10.1017/9781108637435.015`.

27 Dorit S. Hochbaum and David B. Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *J. ACM*, 34(1):144–162, 1987. `doi:10.1145/7531.7535`.

28 Abdolhossein Hoorfar and Mehdi Hassani. Inequalities on the lambert w function and hyperpower function. *J. Inequal. Pure and Appl. Math*, 9(2):5–9, 2008.

29 David R. Karger, Steven J. Phillips, and Eric Torng. A better algorithm for an ancient scheduling problem. *J. Algorithms*, 20(2):400–430, 1996. `doi:10.1006/jagm.1996.0019`.

30 Claire Kenyon. Best-fit bin-packing with random order. In *7th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 359–364. ACM/SIAM, 1996.

31 Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. Primal beats dual on online packing lps in the random-order model. *SIAM J. Comput.*, 47(5):1939–1964, 2018. `doi:10.1137/15M1033708`.

32 Robert D. Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 630–631. SIAM, 2005.

33 Oded Lachish. O(log log rank) competitive ratio for the matroid secretary problem. In *55th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 326–335. IEEE Computer Society, 2014. `doi:10.1109/FOCS.2014.42`.

34 Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1859–1877. SIAM, 2020. `doi:10.1137/1.9781611975994.114`.

35 D. V. Lindley. Dynamic programming and decision theory. *Journal of the Royal Statistical Society*, 10(1):39–51, March 1961.

36 Adam Meyerson. Online facility location. In *42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 426–431. IEEE Computer Society, 2001. `doi:10.1109/SFCS.2001.959917`.

37 Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. In *Beyond the Worst-Case Analysis of Algorithms*, pages 646–662. Cambridge University Press, 2020. `doi:10.1017/9781108637435.037`.

38 Marco Molinaro. Online and random-order load balancing simultaneously. In *28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1638–1650. SIAM, 2017. `doi:10.1137/1.9781611974782.108`.

39 Christopher J. Osborn and Eric Torng. List's worst-average-case or WAC ratio. *J. Sched.*, 11(3):213–215, 2008. `doi:10.1007/s10951-007-0019-7`.

40 John F. Rudin III. *Improved bounds for the on-lijne scheduling problem*. PhD thesis, The University of Texas at Dallas, 2001.

41 Peter Sanders, Naveen Sivadasan, and Martin Skutella. Online scheduling with bounded migration. *Math. Oper. Res.*, 34(2):481–498, 2009. `doi:10.1287/moor.1090.0381`.

42 Martin Skutella and José Verschae. Robust polynomial-time approximation schemes for parallel machine scheduling with job arrivals and departures. *Math. Oper. Res.*, 41(3):991–1021, 2016. `doi:10.1287/moor.2015.0765`.

43 Robert J. Vanderbei. The postdoc variant of the secretary problem. Unpublished. URL: `http://www.princeton.edu/~rvdb/tex/PostdocProblem/PostdocProb.pdf`.

44 Gerhard J. Woeginger. A polynomial-time approximation scheme for maximizing the minimum machine completion time. *Oper. Res. Lett.*, 20(4):149–154, 1997. `doi:10.1016/S0167-6377(96)00055-7`.