Machine Covering in the Random-Order Model

Susanne Albers¹ · Waldo Gálvez² · Maximilian Janke¹

Received: 14 December 2021 / Accepted: 7 July 2022 / Published online: 23 July 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

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 lower bound in the random-order model.

S. Albers, W. Gálvez, M. Janke these authors contributed equally to this work.

Waldo Gálvez
waldo.galvez@uoh.cl
Susanne Albers
albers@in.tum.de
Maximilian Janke
janke@in.tum.de

¹ Department of Computer Science, Technical University of Munich, Boltzmannstrasse 3, 85748 Garching, Germany
² Institute of Engineering Sciences, Universidad de O’Higgins, Avda. Libertador Bernardo O’Higgins 611, 2820000 Rancagua, Chile
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.

**Keywords** Machine Covering · Random-Order · Online Algorithms · Scheduling

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 [42]. The offline problem, also known as Santa-Claus or Max-Min Allocation Problem, has received quite some research interest, see [6, 10, 45] and references therein. In particular, the problem is known to be strongly NP-hard but to allow for a Polynomial-Time Approximation Scheme (PTAS) [45].

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 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 Fig. 1. Indeed, the best possible deterministic algorithm achieves a competitive ratio of \( m \) [45], 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 \( \frac{1}{\sqrt{m}} \).

Such restrictive facts have motivated the study of different semi-online models that provide extra information [7, 14, 35, 38], or extra features [16, 22, 42, 43] to the online algorithm.

Fig. 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 \( m \)
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, 36] and has been applied to a wide variety of problems such as generalized Secretary problems [8, 9, 18, 33, 34], Scheduling problems [2, 3, 39, 40], Packing problems [4, 5, 31, 32], Facility Location problems [37] and Convex Optimization problems [27], among others (see also [26] for a survey chapter). This model 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 $\tilde{O}\left(\sqrt[4]{m}\right)$-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 [28]. The online setting received considerable research attention, and already in 1966, Graham showed that his famous Greedy-strategy is $(2 - 1/m)$-competitive [25]. A long line of research [1, 12, 19, 21, 30] 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 [11, 17, 24], the current best one is 1.88 [41].

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. [42] provide a 2-competitive algorithm for $\beta = 1$. Later results [22, 43] study the interplay between improved competitive ratios and larger values of $\beta$. 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] develop 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 [40]. 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 [39] has studied the Online Load Balancing problem with the objective of minimizing general $l_p$-norms of the machine loads, providing an algorithm that returns solutions of expected norm $(1 + \varepsilon)\text{OPT} + O(\frac{\varepsilon}{m}(m^{1/p} - 1))$ in the random-order model, where OPT denotes the optimal norm. Göbel et al. [23] have studied Average Weighted Completion Time Minimization on a single 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 approach described before 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 it is common in Secretary problems, estimated in a sampling phase, which returns a threshold distinguishing all except for $\sqrt[4]{m}$ 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 \( \tilde{O} (\sqrt{m}) \).

We complement the upper bound of \( \tilde{O} (\sqrt{m}) \) 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 [44]), 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 Sect. 2, we provide the required definitions and tools, and then in Sect. 3, we analyze the Greedy-algorithm in the context of random-order arrival. In Sect. 4, we present our main algorithm and its analysis, and finally in Sect. 5 we introduce the Talent Contest Problem and conclude with a lower bound for the best competitive ratio achievable in the random-order model.

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 \( J_\sigma \) by \( A(J_\sigma) \) and the minimum machine load an optimal offline algorithm may achieve by \( \text{OPT}(J) \) (which is independent of the order \( \sigma \)), one is interested in finding a small (adversarial) competitive ratio \( c = \sup J \sup_{\sigma \in S_n} \frac{A(J_\sigma)}{\text{OPT}(J)} \).

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 \( J \) of size \( n \), we charge a random-order cost \( A_{\text{rom}}(J) = \mathbb{E}_{\sigma \sim S_n}[A(J_\sigma)] = \frac{1}{n!} \sum_{\sigma \in S_n} A(J_\sigma) \). The competitive ratio in the random-order model of \( A \) is \( c = \sup J \frac{A_{\text{rom}}(J)}{\text{OPT}(J)} \). Throughout this work, we will assume that \( n \) is known to the algorithm; this assumption is common in the literature and it can be proved that it does not help in the adversarial setting (see Appendix A for details).

When clear from the context, we will omit the dependency on \( J \).

Given \( 0 \leq i \leq n \), let \( P_i = P_i[J] \) refer to the size of the \( i \)-th largest job in \( J \). 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 \( J_{\text{last}}(M) \) be the last job assigned to machine \( M \). Let \( J_{\text{last}} \) 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 \( ALG + J_{\text{last}}(M) \), as jobs are iteratively assigned to a least loaded machine. Now, remove the \( i - 1 \) largest jobs in \( J_{\text{last}} \) from the solution, and let \( \tilde{L} \) denote the total size of the remaining jobs in \( J_{\text{last}} \). 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

---

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 \( a > 0 \).
\[ L_i \leq m \cdot ALG + (m - i + 1) \cdot P_i. \] Hence, the minimum load achieved by the Greedy-strategy is at least \[ \frac{L_i}{m} - \frac{m}{m} \cdot \frac{ALG + (m - i + 1) \cdot P_i}{m}. \]

With these tools, we can now prove that the Greedy-strategy has a competitive ratio of at most \( O(\frac{m}{H_m}) \) 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.

**Theorem 4** The Greedy-strategy is \((2 + o_m(1)) \frac{m}{H_m}\)-competitive in the random-order model, and furthermore this bound cannot be improved up to the factor of \( 2 + o_m(1) \).

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

For a given order \( J^{\sigma} \) and \( 0 \leq i \leq \hat{k} \), let \( S_i(J^{\sigma}) \) be the total size of small jobs preceded by precisely \( i \) large jobs with respect to \( J^{\sigma} \) (see Fig. 2). We will prove that the minimum load achieved by the Greedy-strategy is at least

\[
\min \left\{ \sum_{i=0}^{\hat{k}} \frac{S_i(J^{\sigma})}{m - i} - \frac{H_m}{2m} \text{OPT}, \frac{H_m}{2m} \text{OPT} \right\}.
\]

A machine is said to be full once it receives a large job; notice that we can assume that no full machine gets assigned further jobs as otherwise the minimum load would be already at least \( \frac{H_m}{2m} \) 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 \( \frac{S_0(J^{\sigma})}{m} \) and, since the upcoming large job gets assigned to a least loaded machine, the average load of the remaining machines is still at least \( \frac{S_0(J^{\sigma})}{m} \). 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 \( \frac{S_0(J^{\sigma})}{m} + \frac{S_1(J^{\sigma})}{m - 1} \). 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 \( \sum_{i=0}^{\hat{k}} \frac{S_i(J^{\sigma})}{m - i} \). 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 \( \sum_{i=0}^{\hat{k}} \frac{S_i(J^{\sigma})}{m - i} - \frac{H_m}{2m} \) OPT.

Now let us understand the random variable \( S(J^{\sigma}) = \sum_{i=0}^{\hat{k}} \frac{S_i(J^{\sigma})}{m - i} \). 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 \( \hat{k} \). Thus, we
Order $J^\sigma$:

- Load $S_0[J^\sigma]$
- Load $S_1[J^\sigma]$
- Load $S_\tilde{k}[J^\sigma]$

![Fig. 2](image)

A possible order $J^\sigma$ 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 $S_i(J^\sigma), i = 0, \ldots, \tilde{k}$

can rewrite $S(J^\sigma) = \sum_{i=0}^{\tilde{k}} \frac{S_i(J^\sigma)}{m-i} = \sum J$ small $\frac{p_J}{m-s(J)}$. Since

$$E \left[ \frac{1}{m-s(J)} \right] = \sum_{i=0}^{\tilde{k}} \frac{1}{k+1} \frac{1}{m-i} = \frac{H_m - H_{m-\tilde{k}}}{\tilde{k}+1},$$

we get by linearity of expectation that

$$E[S(J^\sigma)] = \sum_{J \text{ small}} p_J \frac{H_m - H_{m-\tilde{k}}}{\tilde{k}+1}$$

$$= \frac{H_m - H_{m-\tilde{k}}}{\tilde{k}+1} L_{\tilde{k}+1}$$

$$= \frac{(m-\tilde{k})(H_m - H_{m-\tilde{k}})}{\tilde{k}+1} \frac{L_{\tilde{k}+1}}{m-\tilde{k}}.$$  

Notice that $\frac{L_{\tilde{k}+1}}{m-\tilde{k}}$ is an upper bound for OPT, which we will use later. We will prove that the first factor is decreasing as a function of $\tilde{k} \leq m-1$; being that the case, the expression is at least $\frac{H_m}{m}$, and thus $E[S(J^\sigma)] \geq \frac{H_m}{m} \cdot \frac{L_{\tilde{k}+1}}{m-\tilde{k}}$.

**Claim 5** The expression $\frac{m-\tilde{k}}{k+1} (H_m - H_{m-\tilde{k}})$ is decreasing as a function of $\tilde{k} \leq m-1$.

**Proof** Let $f(\tilde{k}) = \frac{m-\tilde{k}}{k+1} (H_m - H_{m-\tilde{k}})$, and let $k' > \tilde{k}$. Notice first that

$$f(\tilde{k}) = \frac{m-\tilde{k}}{k+1} \sum_{i=\tilde{k}}^{m} \frac{1}{i}$$

$$= \frac{m-\tilde{k}}{k+1} \sum_{i=\tilde{k}}^{m} \frac{1}{i} - \frac{m-\tilde{k}}{k+1} \sum_{i=m-k'}^{\tilde{k}} \frac{1}{i}$$

$$= f(k') \frac{m-k'}{k+1} \frac{1}{m-k'} - \frac{m-\tilde{k}}{k+1} \sum_{i=m-k'}^{\tilde{k}} \frac{1}{i}.$$
Let now \( k' = \tilde{k} + 1 \). From the previous expression we have that

\[
\begin{align*}
  f(\tilde{k}) - f(k') &= f(k') \left( \frac{m - \tilde{k} - 1}{\tilde{k} + 2} \left( \frac{m - \tilde{k} + 2}{m - \tilde{k} - 1} - 1 \right) - \frac{1}{m - \tilde{k} - 1} \right) - \frac{1}{\tilde{k} + 1}
  \\
  &= f(k') \left( \frac{m + 1}{(m - \tilde{k} - 1)(\tilde{k} + 1)} \right) - \frac{1}{\tilde{k} + 1}
  \\
  &= \frac{1}{(m - \tilde{k} - 1)(\tilde{k} + 1)}((m + 1)f(k') - m + \tilde{k})
  \\
  \geq & \quad \frac{1}{(m - \tilde{k} - 1)(\tilde{k} + 1)} \left( \frac{1}{m - \tilde{k} - 1} + \frac{\tilde{k} + 1}{m + 1} \right) - m + \tilde{k}
  \\
  &= 0,
\end{align*}
\]

where in the inequality we used the fact that

\[
f(k') = \frac{m - \tilde{k} - 1}{\tilde{k} + 2} \sum_{i=m-\tilde{k}-1}^{m} \frac{1}{\tilde{k} + 2} \geq \frac{m - \tilde{k} - 1}{\tilde{k} + 2} \left( \frac{1}{m - \tilde{k} - 1} + \frac{\tilde{k} + 1}{m + 1} \right)
\]

With respect to the variance, we get that

\[
\text{Var} \left[ \frac{1}{m - s(J)} \right] \leq \mathbb{E} \left[ \left( \frac{1}{m - s(J)} \right)^2 \right]
\]

\[
= \sum_{i=0}^{\tilde{k}} \frac{1}{\tilde{k} + 1} \left( \frac{1}{m - i} \right)^2
\]

\[
\leq \frac{1}{\tilde{k} + 1} \sum_{i} \frac{1}{i^2} \leq \frac{1}{\tilde{k} + 1} \frac{\pi^2}{6}.
\]

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

\[
\text{Var} \left[ \frac{1}{m - s(J)} \right] \leq \frac{H_m^2}{m} \frac{1}{(m - \tilde{k})^2} \frac{\pi^2}{3}.
\]

Moreover, for two small jobs \( J_i \) and \( J_j \) we have

\[
\text{Cov} \left[ \frac{1}{m - s(J_i)}, \frac{1}{m - s(J_j)} \right] \leq \left( \text{Var} \left[ \frac{1}{m - s(J_i)} \right] \text{Var} \left[ \frac{1}{m - s(J_j)} \right] \right)^{1/2}
\]

\( \copyright \) Springer
\[
\frac{H_2}{m} \leq \frac{1}{m} \frac{\pi^2}{(m - \tilde{k})^2}\]

(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 \(\text{Var}[S(J^\sigma)] = \sum J \text{ small} \frac{p_i p_j}{m - s(J)}\).

\[
\text{Var}[S(J^\sigma)] = \sum_{J_i, J_j \text{ small}} p_i p_j \text{Cov} \left[ \frac{1}{m - s(J_i)}, \frac{1}{m - s(J_j)} \right]
\]

\[
\leq \sum_{J_i \text{ small}} p_i \cdot \sum_{J_j \text{ small}} p_j \cdot \frac{H_2}{m} \frac{1}{m - \tilde{k}} \frac{\pi^2}{3}
\]

\[
= \frac{\pi^2}{3} \frac{H_2}{m} \left( \frac{L_{k+1}}{m - \tilde{k}} \right)^2.
\]

Hence, the standard deviation \(\text{SD}[S(J^\sigma)]\) is at most \(C^{3/2} \frac{L_{k+1}}{m - \tilde{k}}\) for \(C = \sqrt{\frac{\pi^2}{3} \frac{H_2}{m}} = \Theta \left( \frac{\sqrt{(\log(m))^2}}{m} \right)\). Recall that \(L_{k+1} \geq (m - \tilde{k})\text{OPT}\). Chebyshev’s inequality, which allows to bound the probability of deviating from the mean in terms of the standard deviation, yields

\[
P \left[ S(J^\sigma) \leq \left( \frac{H_m}{m} - C\right) \text{OPT} \right] \leq P \left[ S(J^\sigma) \leq \left( \frac{H_m}{m} - C \right) \frac{L_{k+1}}{m - \tilde{k}} \right]
\]

\[
\leq P \left[ |E[S(J^\sigma)] - S(J^\sigma)| \geq C^{-1/2} \cdot \text{SD}[S(J^\sigma)] \right]
\]

\[
\leq C.
\]

We conclude that with probability \(1 - C\) the minimum load achieved by the Greedy-strategy exceeds

\[
\min \left\{ S(J^\sigma) - \frac{H_m}{2m} \text{OPT}, \frac{H_m}{2m} \text{OPT} \right\} \geq \left( \frac{H_m}{2m} - C \right) \text{OPT}
\]

. Thus, its competitive ratio in the random-order model is at most

\[
\frac{1}{1 - C} \frac{H_m}{2m} - C = (2 + o_m(1)) \frac{m}{H_m}
\]

. To prove the lower bound, consider the following instance for \(\varepsilon > 0\): Job set \(J\) consists of \(m - 1\) jobs of size 1 and \(\frac{1}{\varepsilon}\) jobs of size \(\varepsilon\) (see Fig. 3 for a depiction). It is not difficult to see that \(\text{OPT} = 1\) by assigning the jobs of size 1 to different machines and the jobs of size \(\varepsilon\) 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 \(J^\sigma\), the minimum load achieved
by the Greedy-strategy is at most $\sum_{i=0}^{m-1} \frac{S_i(\mathcal{J}_\sigma)}{m-i} + \epsilon m$. If now $\sigma \sim S_n$ is chosen uniformly at random, then the minimum load achieved by the Greedy-strategy is at most $\sum_{i=0}^{m-1} \frac{1}{m(m-i)} + \epsilon m = \frac{H_m}{m} + \epsilon m$; this holds because the probability that a given small job belongs to any $S_i(\mathcal{J}_\sigma)$ is equal to $\frac{1}{m}$, and also due to the fact that the total size of small jobs is equal to 1$^3$. In particular, the competitive ratio in the random-order model of the Greedy-strategy is at most $1/(\frac{H_m}{m} + \epsilon m)$ which approaches $\frac{m}{\epsilon m}$ as $\epsilon \to 0$. □

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 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 $J \in \mathcal{J}$ large if its size is larger than $\text{OPT}[\mathcal{J}] / 100 \sqrt[50]{m}$ and we call it small otherwise. Let $k = k[\mathcal{J}]$ be the number of large jobs in $\mathcal{J}$. Let $L_{\text{small}} = L_{k+1}$ 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 $m - \frac{\sqrt[50]{m^5}}{50}$ large jobs, i.e. $k \leq m - \frac{\sqrt[50]{m^5}}{50}$.

$^3$ We remark that the analysis of the algorithm for this instance is simpler as, for any possible permutation, no job is assigned to a full machine.
Algorithm 1 The online Algorithm for Random-Order Machine Covering

**Input:** Job sequence $\mathcal{J}^0$, $m$ identical parallel machines.

1: Guess $t \in \{-1, 0, 1, \ldots, \left\lceil \frac{3}{4} \log(m) \right\rceil \}$ 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 $(m - 2^t/8 - \sqrt{m}/2)$-th largest job size among these first $n/8$ jobs.

**Phase 2 – Partition.**

6: $\tau \leftarrow 0$

7: for $j = n/8 + 1, \ldots, n$

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 $\tau$ to $p_{\sigma(j)}$ with probability $\frac{1}{2^t \sqrt{m}}$.

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.

Note that the first condition is mostly included for ease of notation; the third condition implies that sequences with $n < m - \frac{\sqrt{m}}{50}$ 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 $\frac{\text{OPT}}{100 \sqrt{m}}$ on simple inputs.

**Proof.** We consider each case from Definition 6 separately:

- If the instance has less than $m$ jobs, then 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 $P_m \geq \frac{\text{OPT}}{100 \sqrt{m}}$ thanks to Lemma 2.
- If there are at most $m - \frac{\sqrt{m}}{50}$ large jobs, then $L_{k+1} \geq \frac{\sqrt{m}}{50} \text{OPT}$ as there are at least $\frac{\sqrt{m}}{50}$ 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 $\frac{\text{OPT}}{100 \sqrt{m}}$.

For an input set $\mathcal{J}$ which is not simple, let $d := \left\lceil \log_2(m - k) \right\rceil$. 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, and consequently from now on we will focus on proper instances of degree $d$; we will show that the corresponding case when $t = d$ in Algorithm 1 returns a $O \left( \sqrt{m} \right)$-approximate solution.
4.2 Algorithm for Proper Inputs of Degree d

Let $J$ 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, while the other machines 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 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 at this point we lack good knowledge about what should be considered small, these jobs will all be assigned to large machines. Let $P^{\uparrow}$ be the element of rank $\frac{m-2d}{8} - \frac{\sqrt{m}}{2}$ 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 $P[ P_{k-8\sqrt{m}-2d} \geq P^{\uparrow} \geq P_k ] \geq \frac{1}{3}$.

**Proof** Let $n_{\text{large}}$ be the number of large jobs among the first $\frac{n}{8}$ jobs in the sequence. Notice that $n_{\text{large}}$ 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 $E_{\sigma \sim S_n}[ n_{\text{large}} ] = \frac{k}{8} \geq \frac{m-2d}{8}$. Moreover

$$\text{Var}[n_{\text{large}}] = \frac{n}{8} \cdot \frac{k}{n} \cdot \frac{n-k}{n} \cdot \frac{7n}{8(n-1)} = \frac{7}{64} \cdot \frac{k(n-k)}{n-1} < \frac{m}{8}.$$ 

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

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

Consider now $n'_{\text{large}}$ to be the number of the $k - 8\sqrt{m} - 2d$ largest jobs among the $\frac{n}{8}$ first jobs in the sequence. Similarly as before, $n'_{\text{large}}$ obeys an hypergeometric distribution with parameters $N = n$, $K = k - 8\sqrt{m} - 2d$ and $r = \frac{n}{8}$, which implies that

$$E_{\sigma \sim S_n}[ n'_{\text{large}} ] = \frac{k - 2d - 8\sqrt{m}}{8} < \frac{m - 2d}{8} - \sqrt{m}.$$
Fig. 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 $\tau$ classifies small jobs. Jobs in between are conservatively classified as large, since misclassifying large jobs is fatal. Increasing $\tau$ due to a small job is a helpful event as less small jobs will be misclassified. On the other hand, increasing $\tau$ 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 (here we use that $k < m$). 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

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

We conclude that

\[
P[P_{k-8\sqrt{m}-2^d} \geq P^\uparrow \geq P_k] = 1 - P[P_{k-8\sqrt{m}-2^d} < P^\uparrow] - P[P^\uparrow < P_k] \geq 1/3.
\]

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 $\tau$, which we initialize as 0. If the incoming job has size at most $\tau$, 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 $\tau$, we schedule $J$ on the least loaded small machine. Else, $J$ is assigned to the least loaded large machine (see Fig. 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 \( \tau \) 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

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

The first inequality is Bernoulli’s inequality, the second one uses that \( d \geq 0 \) and \( m \geq 2 \). 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, properly 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 the set of small jobs scheduled onto small machines is at least \( \frac{31}{800} \sqrt{m} \) \( L_{\text{small}} \), even when conditioned on the input sequence being orderly.

**Proof** Let \( L'_{\text{small}} \) 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

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

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

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

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 \( \frac{1}{2\sqrt{m}} \).

Let \( S \) be the set of small jobs remaining after the sampling phase. We will partition \( S \) into batches of \( 18 \cdot 2^d \sqrt{m} \) 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

\[
18 \cdot 2^d \sqrt{m} \sum_{i \geq 0} \frac{\text{OPT}}{100 \sqrt{m} \cdot 2^i} \leq \frac{18 \cdot 2^d \cdot \text{OPT}}{25} \leq \frac{18 L_{\text{small}}}{25}.
\]
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 $\frac{1}{4 \sqrt{m}}$. Hence, the expected total size of small jobs assigned to small machines is at least

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

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 $\tau$ being increased at large jobs.

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

**Proposition 11** The previously described algorithm is $O\left(\frac{\sqrt{m}}{m}\right)$-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 $P_k \geq \frac{\text{OPT}}{100 \sqrt{m}}$ 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 $\frac{31}{800 \sqrt{m}} L_{\text{small}} - \frac{\text{OPT}}{100 \sqrt{m}}$. The proposition follows from observing that $L_{\text{small}} \geq 2^{d-1} \text{OPT}$ as already argued in the Proof of Lemma 10.

We now have every required ingredient to prove Theorem 1, which we proceed to restate.

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

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 $\tilde{O}(\log(m))$ many possibilities, this guessing induces an extra logarithmic factor on the final competitive ratio. This concludes the proof of the theorem.

## 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 job. 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 [44], but requires additional features particularly catered to our needs.
5.1 The Talent Contest Problem

Consider the following selection problem: \( n \) candidates apply to a yearly talent show contest. 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 global arrival order is chosen uniformly at random. The valuation of each candidate is revealed when the candidate arrives for the first time, and 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 \( \lambda \)-steep if all candidate valuations are guaranteed to be at least by a factor \( \lambda \) apart, i.e. \( \min_{i,j} \frac{v_i}{v_j} \geq \lambda \). It is possible to prove the following bound on the expected value \( P(K, T) \), whose proof we provide in Sect. 5.3.

\begin{equation}
\text{Lemma 12} \quad \text{It holds that } P(K, T) \leq \frac{\zeta(T/2)(T+1)^{T/2}}{\sqrt{2\pi K}} + \frac{4T}{\sqrt{K}}, \quad \text{where } \zeta \text{ is the Riemann Zeta Function. This bound still holds if we restrict ourselves to } \lambda \text{-steep valuations with } \lambda \geq 1.
\end{equation}

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 will 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 \( \lambda \)-steepness is ensured by choosing \( \mu \) 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 algorithm, deterministic or randomized, 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 $\lambda$-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 $K - 1$ largest candidates large, and the $T$ jobs corresponding to the next candidate are called medium. Notice that the size of a medium job is at most $\frac{OPT}{\lambda}$, 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 $T \sum_{i=1}^{\infty} \lambda^{-i} \frac{OPT}{\lambda} = \frac{OPT}{\lambda - 1}$ and are thus called small. They will become negligible for $\lambda \to \infty$.

Consider an online algorithm $A_{MC}$ for Machine Covering in the random-order model. We will derive an algorithm $A_{TC}$ for the Talent Contest problem as follows: $A_{TC}$ 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 $A_{MC}$ contains a machine which has at most $P + 1$ medium jobs and no large job. For this, we consider any fixed input order and, if $A_{MC}$ 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 $A_{TC}$ 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; we set $w_i = 0$ otherwise. Let also $r_i$ be an indicator variable that is 1 if $A_{TC}$ 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 $\mathcal{M}_{small}$ be the machines which do not receive a large job in the schedule of $A_{MC}$, and let $Z_{med}$ be the average number of medium jobs on machines in $\mathcal{M}_{small}$. Our intermediate goal is to show that $Z_{med} < 2 + \sum_{i=2}^{T} w_i$.

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 $|\mathcal{M}_{small}| \geq 1 + \sum_{i=2}^{T} r_i$. Let $d \geq 0$ such that $|\mathcal{M}_{small}| = 1 + d + \sum_{i=2}^{T} r_i$. 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 $\mathcal{M}_{small}$ is at most

$$|\mathcal{M}_{small}| + \sum_{i=2}^{T} (w_i + r_i) = 1 + d + \sum_{i=2}^{T} (w_i + 2r_i).$$

Now, we can bound the average number of medium jobs on $\mathcal{M}_{small}$, 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 \( Z_{\text{med}} \leq 2 + \sum_{i=2}^{T} w_i \). To derive this inequality we assumed that \( Z_{\text{med}} \geq 2 \) but it is trivially true if \( Z_{\text{med}} < 2 \).

Now, let \( z_{\text{med}} \) be the least number of medium jobs on a machine in \( M_{\text{small}} \). Then,

\[ z_{\text{med}} \leq Z_{\text{med}} < 2 + \sum_{i=2}^{T} w_i. \]

Since \( z_{\text{med}} \) 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 \( A_{TC} \) for the Talent Contest problem, we have shown that the schedule of \( A_{MC} \) 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 \( \frac{\text{OPT}}{T-1} \). Thus, machine \( M \) has load at most \( \frac{P+1}{T} \text{OPT} + \frac{\text{OPT}}{T-1} \). In conclusion, the expected load of the least loaded machine in the schedule of \( A_{MC} \) is at most \( \frac{P(K,T+1)}{T} \text{OPT} + \frac{\text{OPT}}{T-1} \), 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 \( \left\lfloor \frac{e^{W(\ln(m))}}{\alpha} - 1 \right\rfloor \) for some fixed constant \( \alpha \geq 1 \). 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

\[ P(K, T) \leq \frac{\zeta(T/2)}{\sqrt{2\pi}} + \frac{4T}{(T+1)^{T/3}} \in O(1). \]

By Lemma 13, no algorithm can be better than \( \left( \frac{T}{c} \right) \)-competitive for \( m = (K-1) \cdot T + 1 < (T+1)^{T+1} \) and some absolute constant \( c \). 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 \( m \mapsto (T+1)^{T+1} \) is \( T \mapsto e^{W(\ln(m))} - 1 \); the second part uses the identity

\[ W(x) \geq \log(x) - \log \log(x) + \frac{1}{2} \frac{\log \log(x)}{\log(x)}, \]

see [29] for more details.
5.3 Proof of Lemma 12

We proceed now with the Proof of Lemma 12. Let us choose the $n$ valuations independently and uniformly at random over the real interval $[0, n]$. For ease of presentation, we are going to consider the reversed problem, where the best candidate is the one having lowest valuation. We can retrieve the original problem by replacing valuation $v$ with $n - v$, thus reverting their order. Consider any algorithm for this problem. The following result is key to prove our desired claim.

Lemma 15 Let $1 \leq h \leq T$, $2/\sqrt{K} \leq \varepsilon < 1/6$ and choose $n$ large enough. Let $P_h$ be the probability that algorithm $A$ wins a point for the $h$-th contest, i.e. it marks the $h$-th arrival of the $K$-th best candidate but not the $h$-th arrival of any better candidate. Then $P_h \leq \left(\frac{1}{\sqrt{2\pi \rho_h^T K}} + 2\varepsilon\right)$, where $ho_h = \frac{T + 1 - h}{T + 1}$. 

As argued later, Lemma 12 follows then by summing up the bound over all possible $h$.

Consider $h$ fixed. We derive Lemma 15 by using $A$ to play a certain game, which is next proven to be difficult. For simplicity, we assume that $N = (1 - \varepsilon)\rho_h^T n$ is an integer (by possibly increasing $n$).

We introduce the binomial guessing game. We are allowed to pick any point $s$ in the interval $[(1 - \varepsilon)K, 2K]$ and any value $g$. Afterwards, $N = (1 - \varepsilon)\rho_h^T n$ random samples are independently drawn from the interval $[0, n]$. We win if precisely $g$ samples are smaller than $s$, otherwise we lose.

Lemma 16 For $n$ large enough, the best probability with which a player can win the binomial guessing game is bounded from above by $\left(\frac{1}{\sqrt{2\pi \rho_h^T K}} + \frac{3\varepsilon}{2}\right)$. 

Proof Let $s$ be the point chosen by the player, and let $X$ be the number of the $N$ random samples that are smaller than $s$. Then, $X$ is binomially distributed, where the probability of success is precisely $s/n$. The optimal strategy of a player for this game is to choose for $g$ the mode, the most likely value, of $X$, which is one of the integers closest to $s/n \cdot N$. By the Poisson Limit Theorem, the probability of $X$ obtaining this (or any!) value is $\frac{e^{-s} s^g}{g!}$ as $n$ and thus $N$ approaches infinity (in this bound we may as well set $g = sN/n$ and ignore the integer rounding, which becomes negligible for $n \to \infty$). For $n$ large enough, the winning probability is thus at most $\frac{e^{-s} s^g}{g!} + \frac{s}{2}$ with $g = sN/n$.

By Stirling’s approximation $\frac{e^{-s} s^g}{g!} + \frac{s}{2} \leq \frac{1}{\sqrt{2\pi s}} + \frac{s}{2}$. Now, substitute $g = sN/n \geq (1 - \varepsilon)K \cdot (1 - \varepsilon)\rho_h^T = (1 - \varepsilon)^2 \rho_h^T K$ to obtain that the probability in the statement of the lemma is at most 

$$\frac{1}{\sqrt{2\pi \rho_h^T K}} + \frac{\varepsilon}{2} = \frac{1}{\sqrt{2\pi \rho_h^T K}} + \frac{\varepsilon}{2} \leq \frac{1}{\sqrt{2\pi \rho_h^T K}} + \varepsilon + \frac{\varepsilon}{2}.$$ 

The last step assumes that $\frac{1}{\sqrt{2\pi \rho_h^T K}} \leq 1 - \frac{3\varepsilon}{2}$, as otherwise the statement of the lemma is trivial. \qed
Now that we introduced the binomial guessing game, we need to set up the stage for $A$ to play on. Consider all the input permutations in which candidates can arrive. We call such an input permutation \emph{regular} if at least $N + 1 = (1 - \varepsilon)\rho_h^T n + 1$ candidates have not yet arrived once the $K$-th candidate has arrived $h$ times, and additionally if the $K$-th best candidate has valuation at least $(1 - \varepsilon)K$ and at most $2K$; otherwise, the input permutation is \emph{irregular}. As the name suggests, we can show that such irregular permutations are very rare.

**Lemma 17** The probability that the input permutation is irregular is smaller than $\varepsilon/2$ for $n$ sufficiently large.

**Proof** Let $L$ be the number of candidates having valuation less than $(1 - \varepsilon)K$. A given candidate has valuation below this value with probability $p = (1 - \varepsilon)\frac{K}{n}$. Thus, $L$ is binomially distributed with parameters $p$ and $n$. We have $E[L] = np = (1 - \varepsilon)K$ and $\text{Var}[L] = np(1 - p) \leq K$. The $K$-th best candidate has valuation at least $(1 - \varepsilon)K$ if $L < K$. Note that $K \geq E[L] + \varepsilon\sqrt{K}\sqrt{\text{Var}[L]}$, and thus Chebyshev’s inequality yields

$$P[L \geq K] \leq P[L \geq E[L] + \varepsilon\sqrt{K}\sqrt{\text{Var}[L]}] \leq \frac{1}{\varepsilon^2 K} \leq \varepsilon/6.$$

For the last inequality, we use $2/\sqrt{K} \leq \varepsilon$.

Let $U$ be the number of candidates that have valuation less than $2K$. The $K$-th best candidate has valuation at most $2K$ if $U > K$. A similar argument as done for $L$ yields $P[U \leq K] \leq \varepsilon/6$.

Finally, consider the correct candidate $W$ of rank $K$ and any other candidate $C$. Candidate $C$ arrives $T$ times, and the probability for each individual arrival to occur after $W$ arrived $h$ times is precisely $\frac{T + 1 - h}{T + 1} = \rho_h$. Thus, the probability that candidate $C$ only arrives after $W$ has arrived $h$ times is $\rho_h^T$. Let $Y$ be the total number of candidates arriving after candidate $W$’s $h$-th arrival. Then, $Y$ is binomially distributed with $E[Y] = \rho_h^T (n - 1)$ and $\sqrt{\text{Var}[Y]} = \sqrt{\rho_h^T (1 - \rho_h^T)(n - 1)} \leq \sqrt{n - 1}$. We choose $n$ large enough such that $(1 - \varepsilon)\rho_h^T n + 1 \leq (1 - \varepsilon/2)\rho_h^T (n - 1)$. This implies that, for $Z = \varepsilon/2 \cdot \rho_h^T$, we have

$$N + 1 = (1 - \varepsilon)\rho_h^T n + 1 \leq (1 - \varepsilon/2)\rho_h^T (n - 1) \leq E[Y] - Z \cdot \sqrt{n - 1} \sqrt{\text{Var}[Y]}.$$

Then again, by Chebyshev’s inequality,

$$P[Y < N + 1] \leq P\left[|Y - E[Y]| \leq Z \cdot \sqrt{n - 1} \cdot \sqrt{\text{Var}[Y]}\right] \leq \frac{1}{Z^2(n - 1)}.$$

This is smaller than $\varepsilon/6$ for $n$ large enough.

The input sequence is irregular only if one of the following holds: $L \geq K$ or $U \leq K$ or $Y < N + 1$. Thus we have shown that the probability of this event is at most $\varepsilon/6 + \varepsilon/6 + \varepsilon/6 = \varepsilon/2$ for $n$ large enough. \hfill $\square$
We use our selection strategy $A$ to play the binomial guessing game. Let $t$ be the smallest time at which precisely $N$ candidates are unknown, and let us assume that the order of our input sequence is regular. Then, the $K$-th candidate has already arrived $h$ times before time $t$. Consider the smallest candidate whose $h$-th arrival has been marked by $A$ by the time $t$, let its valuation and $K - g$ its rank among the currently known candidates. Since we are treating a regular order, $A$ loses unless $(1 - \varepsilon)K \leq s \leq 2K$ and precisely $g$ of the $N$ unknown candidates have valuation below $s$: if there are more than $g$ unknown candidates with valuation smaller than $s$ at that time, then the $K$-th candidate is one of them which is not possible as it already arrived; if there are less, then we cannot earn a point as $s$ would have rank better than $K$ and it is marked. Since the valuations of the $N$ candidates are chosen according to the uniform distribution on $[0, N]$, algorithm $A$ wins a point for the $h$-th contest only if $(s, g)$ wins the binomial guessing game. We can now conclude Lemma 15.

**Proof of Lemma 15** Guess $(s, g)$ according to the previously described strategy at time $t$. If $(s, g)$ is not a valid input to the binomial guessing game, pick any value. We argued previously that algorithm $A$ only wins if $(s, g)$ wins the binomial guessing game or if the input order is irregular. The previous two lemmas bound the probability of the former by $\frac{1}{\sqrt{2\pi \rho_h^T K}} + 3\varepsilon^2$ and the probability of the latter by $\varepsilon^2$.

So far, we have seen that our selection strategy $A$ will win a point for the $h$-th contest with probability at most $\frac{1}{\sqrt{2\pi \rho_h^T K}} + 2\varepsilon$. Thus, in expectation it wins at most $\sum_{h=1}^{T} \left( \frac{1}{\sqrt{2\pi \rho_h^T K}} + 2\varepsilon \right)$ points. Since this holds for any selection strategy, we obtain for $\varepsilon = \frac{2}{\sqrt{K}}$ that

$$P(K, T) \leq \sum_{h=1}^{T} \left( \frac{1}{\sqrt{2\pi \rho_h^T K}} + 2\varepsilon \right)$$

$$= \frac{1}{\sqrt{2\pi K}} \sum_{h=1}^{T} (T + 1)^{T/2 - h} + 2\varepsilon T$$

$$= \frac{(T + 1)^{T/2}}{\sqrt{2\pi K}} \sum_{h=1}^{T} (T + 1)^{-h} + \frac{4T}{\sqrt{K}}$$

$$\leq \frac{(T + 1)^{T/2}}{\sqrt{2\pi K}} \sum_{x=1}^{\infty} x^{-T/2} + \frac{4T}{\sqrt{K}} = \frac{\zeta(T/2)(T + 1)^{T/2}}{\sqrt{2\pi K}} + \frac{4T}{\sqrt{K}}.$$ 

This concludes the proof of the first part of Lemma 12. To see that we can choose the sequence to be $\lambda$-steep simply replace all the values $v_i$ drawn according to the uniform distribution by $\mu^{-v_i}$ for $\mu = \lambda^{n(n-1)/\varepsilon'}$ for some $\varepsilon'$. Since an online algorithm can also compute this operation, the previous bound still applies to this modified sequence. Moreover, this sequence is almost guaranteed to be $\lambda$-steep.

**Lemma 18** The probability that the resulting valuation is not $\lambda$-steep is less than $\varepsilon'$. 

\[\text{Springer}\]
Proof One can easily verify that the valuation is $\lambda$-steep if (and only if) no two of the $v_i$ picked according to the uniform distribution have distance less than $d = \frac{\epsilon'}{n(n-1)}$. Consider $1 \leq i \leq n$ and assume that we first pick valuation $v_1 \in [0, n]$, then $v_2$ and so on. Then the probability that valuation $v_i$ has distance less than $d$ from any of the previously picked valuations is at most $(i - 1) \cdot 2d$. Thus, the probability that there is a pair of valuations at distance less than $d$ is at most $\sum_{i=1}^{n} (i - 1) \cdot 2d = \frac{n(n-1)}{2} \cdot 2d = \epsilon'$.

Thus, we can pick a valuation according to the previous distribution and then replace valuation $v_i$ by $\mu - v_i$. If this valuation is not $\lambda$-steep we pick any $\lambda$-steep valuation instead. Since the probability of this latter case is at most $\epsilon'$ the probability of the online algorithm being right increases by at most $\epsilon'$. The second part of Lemma 12 follows since we can choose $\epsilon'$ arbitrarily small, concluding the proof.

6 Conclusion

In this work we have studied the Machine Covering problem in the Random-Order model and provided improved competitive algorithms as well as lower bounds for the best possible competitive ratio achievable in this model. The first open question is of course to narrow this gap, either by improving the lower bound of $\omega(\log n/(\log \log n))$ or by devising algorithms with a competitive ratio better than $\tilde{O}(m^{1/4})$. Another interesting question is to study related assignment problems in the Random-Order model such as Scheduling with Restricted Assignments or the case of Related Machines, where problematic instances seem again to rely on hiding classes of jobs at the end of the corresponding sequences. Finally, it would be interesting to see if our techniques can be applied to obtain lower bounds for other problems under the Random-Order model, which seems to be in general an elusive task.

Acknowledgements We thank the anonymous reviewers for helpful comments that significantly improved the presentation of the results.

A Knowing $n$ Does not Help in the Adversarial Model.

As common in the area of online algorithms, we ignore computability issues. Formally, a deterministic online algorithm that does not know $n$ is a function that receives as input a sequence of jobs $J_1, \ldots, J_n$, which corresponds to the information the algorithm has at the moment of scheduling $J_n$, and returns a machine $M$ onto which the job $J_n$ is scheduled. Similarly, a deterministic online algorithm that knows $n$ receives as input a tuple $n, J_1, \ldots, J_n$ and returns the machine $M$ onto which job $J_n$ is scheduled. Here, $n \geq n'$ is the total length of the input. Randomized algorithms are simply probability distributions over deterministic algorithms.

Proposition 19 Let $A$ be any (possibly randomized) online algorithm that is $c$-competitive in the adversarial model knowing $n$ beforehand. Then there is a strategy
that is $c$-competitive not knowing $n$. If $A$ is deterministic, the strategy $B$ can be chosen to be deterministic too.

**Proof** Let us denote the algorithm $A$ that knows beforehand that the input will have length $n$ by $A[n]$. This algorithm outputs schedules for any sequence of jobs having length at most $n$. Given some number $n' \in \mathbb{N}$, let $\mathcal{S}[n']$ be the set of all probability distributions over possible schedules of $n'$ jobs. Schedules of $n'$ jobs are nothing but assignments $\{1, \ldots, n'\} \rightarrow \{1, \ldots, m\}$. Let $F$ be the set of all such maps. We can then identify $\mathcal{S}[n']$ with a compact subspace of $[0, 1]^F \subset \mathbb{R}^F$. In fact it is the standard simplex spanned by the set $\mathcal{E}[n'] \subset \mathcal{S}[n']$ consisting of probability distributions that choose one schedule with probability 1, or, geometrically speaking, the set of standard base vectors in $\mathbb{R}^M$. For $n' > 0$, forgetting how we scheduled the last job yields a canonical continuous map $r_{n'}: \mathcal{S}[n'] \rightarrow \mathcal{S}[n'-1]$.

Given jobs $J_1, \ldots, J_n$ and $n \geq n'$, let $s^n_{n'} = s^n_{n'}(J_1, \ldots, J_n) \in \mathcal{S}[n']$ be the probability distribution of schedules obtained by algorithm $A[n]$ on the job sequence $J_1, \ldots, J_n$. Consider the set of points $s^n_{n'}$ with $n \geq n'$. We pick inductively a limit point $s_{n'}$ of this point set such that $r_{n'}(s_{n'}) = s_{n'-1}$ (this is possible because $r_{n'}$ is continuous). The choice of $s_{n'}$ only depends on the jobs $J_1, \ldots, J_{n'}$, hence can be picked ‘online’. Now all our algorithm $B$ has to do is to maintain its schedule at time $n'$ according to the probability distribution $s_{n'}$. The condition $r_{n'}(s_{n'}) = s_{n'-1}$ ensures that this is possible.

We will prove that this algorithm is $c$-competitive. Pick $\varepsilon > 0$ and any input sequence $J_1, \ldots, J_{n'}$. As $s_{n'}$ is a limit point of the set of points $s^n_{n'}$, there exists $n \geq n'$ such that $(1-\varepsilon)s^n_{n'} \leq s_{n'}$. In other words, if algorithm $A[n]$ obtains schedule $F$ after treating $J_1, \ldots, J_{n'}$ with probability $p$, then $B$ obtains the same schedule with probability at least $(1-\varepsilon)p$. In particular, the expected minimum load of $B$ on $J_1, \ldots, J_{n'}$ is at least $(1-\varepsilon)$ times the minimum load of $A[n]$ on that sequence. The latter minimum load is at least $c \cdot \text{OPT}(J_1, \ldots, J_{n'})$ since $A[n]$ is $c$-competitive. We have thus shown that the expected minimum load of $B$ is at least $(1-\varepsilon)c \cdot \text{OPT}$. The proposition follows by taking the limit $\varepsilon \rightarrow 0$.

If $A$ is deterministic, all the points $s^n_{n'}$ lie in the closed, in fact discrete, subset $\mathcal{E}[n'] \subset \mathcal{S}[n']$ of probability distributions simply picking one schedule with probability 1. All limit points, in particular the points $s_{n'}$, lie in $\mathcal{E}[n']$ too. Thus, $B$ will also be deterministic.

**References**

1. Albers, S.: Better bounds for online scheduling. SIAM J. Comput. 29(2), 459–473 (1999). https://doi.org/10.1137/S0097539797324874
2. Albers, S., Janke, M.: Scheduling in the random-order model. In: ICALP, vol 168. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, pp. 68:1–68:18, (2020) https://doi.org/10.4230/LIPIcs.ICALP.2020.68
3. Albers, S., Janke, M.: Scheduling in the Secretary Model. In: FSTTCS, vol 213. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, pp. 6:1–6:22 (2021) https://doi.org/10.4230/LIPIcs.FSTTCS.2021.6
4. Albers, S., Khan, A., Ladewig, L.: Best fit bin packing with random order revisited. Algorithmica 83(9), 2833–2858 (2021). https://doi.org/10.1007/s00453-021-00844-5
5. Albers, S., Khan, A., Ladewig, L.: Improved online algorithms for knapsack and GAP in the random order model. Algorithmica 83(6), 1750–1785 (2021). https://doi.org/10.1007/s00453-021-00801-2
6. Asadpour, A., Saberi, A.: An approximation algorithm for max-min fair allocation of indivisible goods. SIAM J. Comput. 39(7), 2970–2989 (2010). https://doi.org/10.1137/080723491
7. Azar, Y., Epstein, L.: On-line machine covering. Journal of Scheduling 1(2), 67–77 (1998). https://doi.org/10.1002/(SICI)1099-1425(199808)1:2<67::AID-JOS6>3.0.CO;2-Y
8. Babai, M., Immorlica, N., Kempe, D., et al: A knapsack secretary problem with applications. In: APPROX/RANDOM, vol. 4627, pp. 16–28. Springer (2007) https://doi.org/10.1007/978-3-540-74208-1_2
9. Babai, M., Immorlica, N., Kempe, D., et al.: Matroid secretary problems. J. ACM 65(6), 35:1-35:26 (2018). https://doi.org/10.1145/3122512
10. Bansal, N., Sviridenko, M.: The santa claus problem. In: STOC. ACM, pp. 31–40. (2006) https://doi.org/10.1145/1132516.1132522
11. Bartal, Y., Karloff, H.J., Rabani, Y.: A better lower bound for on-line scheduling. Inf. Process. Lett. 50(3), 113–116 (1994). https://doi.org/10.1016/0020-0190(94)00026-3
12. Bartal, Y., Fiat, A., Karloff, H.J., et al.: New algorithms for an ancient scheduling problem. J. Comput. Syst. Sci. 51(3), 359–366 (1995). https://doi.org/10.1006/jcss.1995.1074
13. Dynkin, E.B.: The optimum choice of the instant for stopping a Markov process. Soviet Math. Dokl. 4, 627–629 (1962)
14. Ebenlendr, T., Noga, J., Sgall, J., et al.: A note on semi-online machine covering. In: W AOA, vol 3879, pp. 110–118. Springer (2005) https://doi.org/10.1007/11671411_9
15. Epstein, L.: A survey on makespan minimization in semi-online environments. J. Sched. 21(3), 269–284 (2018). https://doi.org/10.1007/s10951-018-0567-z
16. Epstein, L., Levin, A., van Stee, R.: Max-min online allocations with a reordering buffer. SIAM J. Discret. Math. 25(3), 1230–1250 (2011). https://doi.org/10.1137/100794006
17. Faigle, U., Kern, W., Turán, G.: On the performance of on-line algorithms for partition problems. Acta Cybern 9(2), 107–119 (1989)
18. Feldman, M., Svensson, O., Zenklusen, R.: A simple O(log log(rank))-competitive algorithm for the matroid secretary problem. Math. Oper. Res. 43(2), 638–650 (2018). https://doi.org/10.1287/moor.2017.0876
19. Fleischer, R., Wahl, M.: Online scheduling revisited. In: ESA, vol 1879, pp. 202–210. Springer (2000) https://doi.org/10.1007/3-540-45253-2_19
20. Friesen, D.K., Deuermeyer, B.L.: Analysis of greedy solutions for a replacement part sequencing problem. Math. Oper. Res. 6(1), 74–87 (1981). https://doi.org/10.1287/moor.6.1.74
21. Galambos, G., Woeginger, G.J.: An on-line scheduling heuristic with better worst case ratio than graham’s list scheduling. SIAM J. Comput. 22(2), 349–355 (1993). https://doi.org/10.1137/0222026
22. Gálvez, W., Soto, J.A., Verschae, J.: Symmetry exploitation for online machine covering with bounded migration. ACM Trans. Algorithms 16(4), 43:1-43:22 (2020). https://doi.org/10.1145/3397535
23. Göbel, O., Kesselheim, T., Tönnis, A.: Online appointment scheduling in the random order model. In: ESA, vol 9294, pp. 680–692. Springer (2015) https://doi.org/10.1007/978-3-662-48350-3_57
24. Gormley, T., Reingold, N., Torng, E., et al.: Generating adversaries for request-answer games. In: SODA. ACM/SIAM, pp. 564–565 (2000)
25. Graham, R.L.: Bounds for certain multiprocessor anomalies. The Bell System Technical Journal 45(9), 1563–1581 (1966). https://doi.org/10.1002/j.1538-7305.1966.tb01709.x
26. Gupta, A., Singla, S.: Random-order Models. In: Beyond the Worst-Case Analysis of Algorithms, pp. 234–258. Cambridge University Press, Cambridge (2020). https://doi.org/10.1007/978-1108637435.015
27. Gupta, A., Mehta, R., Molinaro, M.: Maximizing profit with convex costs in the random-order model. In: ICALP, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, vol. 107, pp. 71:1–71:14. (2018) https://doi.org/10.4230/LIPIcs.ICALP.2018.71
28. Hochbaum, D.S., Shmoys, D.B.: Using dual approximation algorithms for scheduling problems theoretical and practical results. J. ACM 34(1), 144–162 (1987). https://doi.org/10.1145/7531.7535
29. Hoofar, A., Hassani, M.: Inequalities on the lambert w function and hyperpower function. J. Inequal Pure Appl. Math. 9(2), 5–9 (2008)
30. Karger, D.R., Phillips, S.J., Torng, E.: A better algorithm for an ancient scheduling problem. J. Algorithms 20(2), 400–430 (1996). https://doi.org/10.1006/jagm.1996.0019
31. Kenyon, C.: Best-fit bin-packing with random order. In: SODA. ACM/SIAM, pp. 359–364 (1996)
32. Kesselheim, T., Radke, K., Tönnis, A., et al.: Primal beats dual on online packing lps in the random-order model. SIAM J. Comput. 47(5), 1939–1964 (2018). https://doi.org/10.1137/15M1033708
33. Kleinberg, R.D.: A multiple-choice secretary algorithm with applications to online auctions. In: SODA. SIAM, pp. 630–631 (2005)
34. Lachish, O.: $O(\log \log \text{rank})$ competitive ratio for the matroid secretary problem. In: FOCS. IEEE Computer Society, pp. 326–335 (2014) https://doi.org/10.1109/FOCS.2014.42
35. Lattanzi, S., Lavastida, T., Moseley, B., et al.: Online scheduling via learned weights. In: SODA. SIAM, pp. 1859–1877 (2020) https://doi.org/10.1137/1.9781611975994.114
36. Lindley, D.V.: Dynamic programming and decision theory. J. Roy. Stat. Soc. 10(1), 39–51 (1961)
37. Meyerson, A.: Online facility location. In: FOCS. IEEE Computer Society, pp. 426–431 (2001) https://doi.org/10.1109/SFCS.2001.959917
38. Mitzenmacher, M., Vassilvitskii, S.: Algorithms with Predictions. In: Beyond the Worst-Case Analysis of Algorithms, pp. 646–662. Cambridge University Press, Cambridge (2020). https://doi.org/10.1017/9781108637435.037
39. Molinaro, M.: Online and random-order load balancing simultaneously. In: SODA. SIAM, pp. 1638–1650 (2017) https://doi.org/10.1137/1.9781611974782.108
40. Osborn, C.J., Torng, E.: List’s worst-average-case or WAC ratio. J. Sched. 11(3), 213–215 (2008). https://doi.org/10.1007/s10951-007-0019-7
41. Rudin III, J.F.: Improved bounds for the on-line scheduling problem. PhD thesis, The University of Texas at Dallas (2001)
42. Sanders, P., Sivadasan, N., Skutella, M.: Online scheduling with bounded migration. Math. Oper. Res. 34(2), 481–498 (2009). https://doi.org/10.1287/moor.1090.0381
43. Skutella, M., Verschae, J.: Robust polynomial-time approximation schemes for parallel machine scheduling with job arrivals and departures. Math. Oper. Res. 41(3), 991–1021 (2016). https://doi.org/10.1287/moor.2015.0765
44. Vanderbei, R.J.: The postdoc variant of the secretary problem. http://www.princeton.edu/~rvdb/tex/PostdocProblem/PostdocProb.pdf, Unpublished (1983)
45. Woeginger, G.J.: A polynomial-time approximation scheme for maximizing the minimum machine completion time. Oper. Res. Lett. 20(4), 149–154 (1997). https://doi.org/10.1016/S0167-6377(96)00055-7

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.