Random-Order Interval Selection

Allan Borodin
bor@cs.toronto.edu

Christodoulos Karavasilis
ckar@cs.toronto.edu

Abstract

In the problem of online unweighted interval selection, the objective is to maximize the number of non-conflicting intervals accepted by the algorithm. In the conventional online model of irrevocable decisions, there is an $\Omega(n)$ lower bound on the competitive ratio, even for randomized algorithms [1]. In a line of work that allows for revocable acceptances, Faigle and Nawijn [8] gave a greedy 1-competitive (i.e. optimal) algorithm in the real-time model, where intervals arrive in order of non-decreasing starting times. The natural extension of their algorithm in the adversarial (any-order) model is $2k$-competitive [2], when there are at most k different interval lengths, and that is optimal for all deterministic, and memoryless randomized algorithms. We study this problem in the random-order model, where the adversary chooses the instance, but the online sequence is a uniformly random permutation of the items. We consider the same algorithm that is optimal in the cases of the real-time and any-order models, and give an upper bound of 2.5 on the competitive ratio under random-order arrivals.

We also show how to utilize random-order arrivals to extract a random bit with a worst case bias of $\frac{2}{3}$, when there are at least two distinct item types. We use this bit to derandomize the barely random algorithm of Fung et al. [10] and get a deterministic 3-competitive algorithm for single-length interval selection with arbitrary weights.

1 Introduction

In the problem of interval scheduling on a single machine, there is a set of intervals on the real line, each with a fixed starting time and end time, and we must choose a subset of non-conflicting intervals. In the unweighted setting, the goal is to maximize the cardinality of the subset. In terms of the objective function, this is equivalent to finding a maximum independent set of an interval graph. In weighted variations, each interval is associated with a weight, and we aim to maximize the total weight of the solution. In the online version of the problem, intervals arrive one at a time, and the algorithm must either accept an interval, or forever discard it. Following existing work on interval scheduling, we consider a model where any new interval can be accepted, displacing any conflicting intervals currently in the solution. Displaced intervals, similar to intervals that are rejected upon arrival, can never be taken again. While competitive analysis is traditionally concerned with *irrevocable* decisions, that assumption is sometimes relaxed in order to tackle cases where results are overly pessimistic, or if the application permits it. For example, for the problem of unweighted interval selection, in the real-time model where intervals arrive in order of increasing starting times, there is an $\Omega(n)$ lower bound, even for randomized algorithms [1]. Different types of revocable decisions are problem-specific, and appear under various names, such as *preemption*, *replacement*, *free disposal*, and *recourse*. Other examples of problems that have been studied under such relaxed models are the knapsack problem [18], submodular maximization [4], weighted matching [9], maximum coverage [24], and other graph problems [3]. It is worth noting that algorithms in these models are also relevant when online algorithms are used to construct offline solutions, where revoking decisions does not violate the model and may come at no additional cost.

In the adversarial model of online algorithms, the optimal deterministic algorithm for unweighted interval selection with revoking is $2k$-competitive [2], where k is the number of different interval lengths. We study this problem under random-order arrivals, a model used for beyond worst-case analysis that also captures stochastic i.i.d. settings (see Gupta & Singla [14]). While there are many instances where random arrivals help, there are problems where the competitive ratio is not significantly improved (e.g. Steiner trees where a greedy algorithm is $O(\log n)$-competitive in the worst-case, and there are $\Omega(\log n)$ bounds for both adversarial and random-order arrivals [14]). We show that the simple greedy algorithm that is optimal $2k$-competitive in the adversarial case, is 2.5-competitive in the random-order model, removing the dependence on k. In this model, the adversary chooses the input items, but the online sequence is a uniformly random permutation of the items. Finally, we use the application of interval scheduling as motivation to begin to understand a very general issue in online algorithms, namely to understand the power of randomized algorithms with adversarial arrival order compared to deterministic algorithms with random arrivals. In this regard, we are interested to what extent can we extract random bits from the randomness in the arrival order. Specifically, we show how to take advantage of the randomness in the arrival order to extract a random bit with bounded bias.

Some examples of applications related to interval scheduling are routing [23], computer wiring [15], project selections during space missions [16], and satellite photography [11]. We refer the reader to the surveys by Kolen et al. [19] and Kovalyov et al. [20] for a more detailed discussion on the applications of interval scheduling.

Related Work. Lipton and Tomkins [21] introduced the problem of online interval scheduling. They consider the real-time setting, proportional weights, and do not allow for displacement of intervals in the solution. They give a randomized algorithm that is $O((\log \Delta)^{1+\epsilon})$-competitive, where Δ is the ratio of the longest to shortest interval. In the real-time unweighted setting, Faigle and Nawijn [8] consider a simple greedy 1-competitive deterministic algorithm with revoking. Without revoking, there is an $\Omega(n)$ lower bound both for deterministic and randomized [1] algorithms. Woeginger [27] considers a real-time, weighted variation of the problem with revoking, and shows that no deterministic algorithm can be constant competitive for general weights. Canetti and Irani [5] extend this impossibility to randomized algorithms with revoking. When an interval's weight is a function of its length, Woeginger gives an optimal 4-competitive deterministic algorithm for special classes of weight functions. Randomized algorithms were considered for these special classes of functions [26, 7], with Fung et al. [10] currently having the best known upper bound of 2.

In the adversarial model, or *any-order* arrivals, Bachmann et al. [1] show a lower bound of $\Omega(n)$ for randomized algorithms in the offline unweighted setting without revoking. Borodin and Karavasilis [2] consider the unweighted problem with revoking, and give an optimal $2k$-competitive deterministic algorithm, where k is the number of different interval lengths. This algorithm is a natural extension of the algorithm by Faigle and Nawijn [8] for any-order arrivals. Emek et al. [6] give a randomized algorithm that is 6-competitive for unweighted interval selection. For the case of proportional weights with revoking, Garay et al. [12] give an optimal $(2+\sqrt{5}) \approx 4.23$-competitive deterministic algorithm for the problem of call control on the line, which also applies to any-order interval selection. The 2-competitive randomized algorithm by Fung et al. [10] for the case of real-time, single-length, arbitrary weights, also applies to the any-order case.

In the random-order setting, Im and Wang [17] consider the interval scheduling secretary prob-

lem, where weighted jobs have to be processed within some interval, not necessarily continuously. They give a $O(\log D)$-competitive randomized algorithm, where D is the maximum interval length of any job. More relevant to our setting, Borodin and Karavasilis [2] consider single-length unweighted interval selection with random arrivals, and show that the only deterministic memoryless algorithm that may be better than 2-competitive, is a *one-way* algorithm that replaces intervals in the same direction. Garg et al. [13] consider interval scheduling and maximum independent set of hyperrectangles under random arrivals. They do not allow for revoking of accepted intervals, and give a non-greedy algorithm that is *strongly* (a form of high probability) $O(\log n \cdot \log \log n)$-competitive for interval selection. We note that their algorithm requires knowledge of n, the size of the input instance. Furthermore, they show that no algorithm that is not provided n can be strongly $O(n^{1-\epsilon})$-competitive, for all $\epsilon > 0$.

Our Results. We consider the optimal, simple greedy algorithm of [2] that is $2k$-competitive in the adversarial any-order setting, and extends the 1-competitive algorithm of [8] from the real-time setting. We analyze that algorithm under uniformly random arrivals, and we give an upper bound of 2.5 on the competitive ratio. We use a charging argument motivated by [2] and bound the competitive ratio by the expected amount of maximum charge on any interval. We also give a lower bound of $\frac{12}{11}$ on the competitive ratio of all deterministic algorithms with revoking under random arrivals (appendix B). This bound separates the random-order model with the real-time model, where 1-competitiveness is attainable.

Furthermore, we utilize the random arrival of online items to extract a random bit with worst case bias of $\frac{2}{3}$, when there are at least two distinct item types. We use this bit to derandomize the barely random algorithm by Fung et al. [10] in the case of single-length arbitrary weights. This technique may be applied to other *classify and randomly select* algorithms that choose between two classes of items, when revoking is allowed. We also consider a setting where there exists a global ordering amongst all input items. For example, this could apply to interval scheduling under the assumption that all intervals have distinct starting times. This setting allows for unbiased bits to be extracted throughout the execution of the algorithm, and may be useful for choosing amongst multiple classes of input items. Under this assumption, we give a 6-competitive algorithm in the case of two different interval lengths and arbitrary weights.

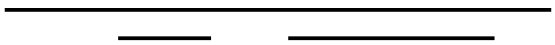
Organization of the paper. Section 2 includes definitions and a description of how the mapping from optimal intervals to intervals accepted by the algorithm is defined. We also show how the competitive ratio is bounded. Section 3 contains the main analysis of the algorithm in the random-order model. In Section 3.1 we deal with the case of two interval lengths ($k = 2$). We explore the dynamics of redefining the mapping because of the displacement of intervals (revoking), and this analysis is later used to show the general case for any $k > 2$ in Section 3.2. Section 4 presents two different processes to extract random bits using random-order arrivals, which we use to derandomize a 1-random-bit algorithm for the case of single-length arbitrary weights intervals. We end with some conclusions and open problems.

2 Preliminaries

The model consists of intervals arriving on the real line. An interval I_i is specified by a starting point s_i, and an end point f_i, with $s_i < f_i$. It occupies space $[s_i, f_i)$ on the line, and the conventional notions of intersection, disjointness, and containment apply. There are two main ways intervals

can conflict, and they are shown in figure 1. One type of conflict is a *partial conflict*, and the other type is *inclusion*, or *containment*. In the case of containment, we say that the smaller intervals are *subsumed* by the larger one. We use k to denote the number of different interval lengths of an instance. An instance with k different lengths, can have a *nesting depth* of at most $k-1$.

(a) Partial Conflict.



(b) Containment with nesting depth 1.

Figure 1: Types of conflicts.

Let OPT denote the size of an optimal solution, and ALG the size of the algorithm's solution. We will also use OPT and ALG to refer respectively to an optimal solution, and the solution returned by the algorithm. The meaning should always be clear from context. We use the notion of competitive ratio to measure the performance of an online algorithm. Given an algorithm A (creating a solution ALG), we consider the strict competitive ratio of A : $CR(A) = \max_{\mathcal{I}} \frac{OPT}{\mathbb{E}[ALG]}$, where the expectation is over all the permutations of the input instance, and the maximum is over all input instances.

In our proofs, we make use of a charging argument. We will now describe how the charging is done. Given an instance (set of intervals) $\mathcal{I}$ and an interval arrival sequence σ, we choose an optimal solution $OPT_\sigma^{\mathcal{I}}$, and define a mapping $\mathcal{F}_\sigma^{\mathcal{I}} : OPT_\sigma^{\mathcal{I}} \to ALG_\sigma^{\mathcal{I}}$ that shows how the intervals from an optimal solution are charged to intervals taken by the algorithm. The mapping $\mathcal{F}_\sigma^{\mathcal{I}}$ can be viewed as being formed and redefined throughout the execution of the algorithm as follows: On the arrival of interval $I \in OPT_\sigma^{\mathcal{I}}$, if I is taken by the algorithm, it is mapped onto itself. If I is rejected because it conflicts with some intervals taken by the algorithm, it is arbitrarily mapped to one of those conflicting intervals. Whenever an interval I' is taken by replacing an existing interval I'', all optimal intervals mapped to I'' up to that point, will then be mapped to I'. These two first cases where optimal intervals are charged upon arrival, are instances of *direct charging*. Whenever an interval is replaced by another, an instance of *transfer charging* occurs to the new interval. Notice that in the end, every interval $I \in OPT_\sigma^{\mathcal{I}}$ is mapped to exactly one interval taken by the algorithm. We note that being able to choose a different OPT for a given sequence σ, provides flexibility and facilitates our proofs. This may be important in tackling other problems in the random-order model, especially when revoking is allowed.

Given the mapping $\mathcal{F}_\sigma^{\mathcal{I}}$, let $\Phi : ALG \to \mathbb{Z}_{\geq 0}$ denote the charging function, which shows, at any time during the execution, the total amount of charge to any interval currently in the online algorithm's solution. That is, $\Phi(I) = |\{I' \in OPT : \mathcal{F}(I') = I\}|$. We can also express the amount of charge as $\Phi(I) = TC(I) + DC(I)$, where $TC(I)$ denotes the total amount of transfer charge to I at the time it was taken by the algorithm, and $DC(I)$ denotes the total amount of direct charge to I.

Notice how at the end of the execution, $\sum_{I \in ALG} \Phi(I) = OPT$. We can now bound the competitive ratio of an algorithm for any instance as follows:

$$\frac{OPT}{\mathbb{E}[ALG]} = \frac{\mathbb{E}\left[\sum_{1 \le i \le ALG} \Phi(I_i)\right]}{\mathbb{E}[ALG]}$$

$$\le \frac{\mathbb{E}[ALG] \max_I \{\mathbb{E}[\Phi(I) \mid I \in ALG]\}}{\mathbb{E}[ALG]}$$

$$= \max_I \{\mathbb{E}[\Phi(I) \mid I \in ALG]\}$$

The first equality is because the sum $\Phi(I_1) + ... + \Phi(I_{ALG})$ is always equal to OPT, which is a constant determined by the instance $\mathcal{I}$, and does not depend on the random arrival sequence. The inequality holds by applying Wald's inequality (as given in Young [28], lemma 4.1). It follows that it suffices to bound the expected charge on every interval in ALG.

Definition 2.1 (Predecessor trace)**.** Let I be an interval in the algorithm's final solution. The predecessor trace of I is the maximal list of intervals $(P_1, P_2, ..., P_k = I)$ such that P_i was at some point accepted by the algorithm, but was later replaced by P_{i+1}.

A predecessor trace is analogous to Woeginger's [27] predecessor chain in the real-time model.

3 Main Analysis for the Random-Order Model

In this section we analyze the performance of Algorithm 1. This algorithm is *greedy*, in the sense that when an arriving interval does not conflict with anything, it is always accepted by the algorithm. If there are conflicts, a new interval is only accepted if it is entirely subsumed by an interval currently in the solution, which in turn gets replaced. Notice that an interval taken (maybe temporarily) by this algorithm can be directly charged by at most two optimal intervals. This is because any interval can partially conflict with at most two intervals from an optimal solution. This fact is also relevant for single-length instances ($k = 1$), where no interval is replaced by this algorithm. In that case we have $TC(I) = 0$ and $DC(I) \le 2$ for every interval I, giving us an upper bound of 2 on the competitive ratio. A lower bound of $\frac{2n}{n+2}$ is given in figure 2 ($ALG = 1$ w.p. $\frac{n-2}{n}$, $OPT = 2$).

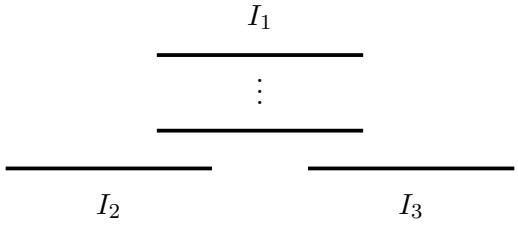


Figure 2: Instance where Algorithm 1 is $\frac{2n}{n+2}$-competitive.

Algorithm 1

On the arrival of I:
$I_s \leftarrow$ Set of intervals currently in the solution conflicting with I
if $I_s = \emptyset$ **then**
    Take I
    return
for $I' \in I_s$ **do**
    if $I \subset I'$ **then**
        Take I and discard I'
        return
Discard I

We will now study the case of only two interval lengths. The results of this section will later be used to show the result for $k > 2$.

3.1 Case of $k = 2$

We first focus on a *base instance* that showcases the dynamics of transfer charging. Note that in this case, any predecessor trace is of length at most two. Consider an instance with two different lengths as shown in figure 3. Let L, R, M, S denote the sets of corresponding intervals. The set S of small intervals is entirely contained in the large intervals of M, and we make no assumptions about the structure of S. In fact, intervals in S are also allowed to partially[1] conflict with intervals in $L \cup R$. An optimal solution consists of intervals $I_L \in L$, $I_R \in R$, and some intervals $I_s \subseteq S$. For the purposes of charging, we will be choosing the optimal solution that contains the latest arriving $I_L \in L$ and $I_R \in R$. The intervals in L and R are depicted as small intervals, but in reality they could be either small or large. We also note that intervals that are depicted as copies do not have to perfectly coincide.

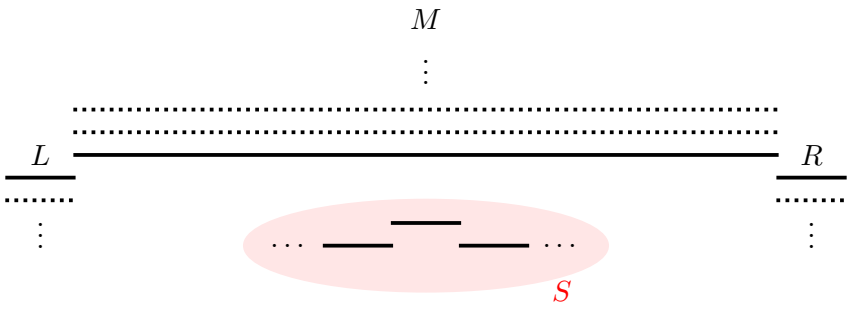


Figure 3: Base instance for transfer charging with $k = 2$.

Lemma 3.1. *For any instance with a structure as depicted in figure 3, we have that* $\mathbb{E}[\Phi(I) \mid I \in ALG] \leq 2.5$.

Proof. We will be writing $\mathbb{E}[\Phi(I)]$ for readability. We have that $\Phi(I) = TC(I) + DC(I)$. As mentioned before, $\forall I, DC(I) \leq 2$. We will now bound $\mathbb{E}[TC(I)]$. Let TC_1 denote the event that a transfer charge of 1 occurs, and TC_2 denote the event that a transfer charge of 2 occurs. We

[1]W.l.o.g. no interval in $I_s \in S$ is entirely contained in an interval in $L \cup R$. If that was the case, I_s would be considered optimal and at least one of L, R would be empty.

focus on the first arrival of an interval from S, as that interval will receive the transfer charge. Let $N = |L| + |R| + |M| + |S|$.

We want to compute: $\underset{|L|,|R|,|M|,|S|}{argmax}(\mathbb{E}[TC(I)])$, where $\mathbb{E}[TC(I)] = Pr(TC_1) + 2Pr(TC_2)$.

Case of TC_2: For a transfer charge of 2 to occur, it must be that an interval from M arrives first, and that all the intervals in $L \cup R$ arrive before the first interval from S. This is an experiment of drawing without replacement, and the probability that we get all intervals in $L \cup R$ before the first interval from S is the following:

$$\frac{|L| + |R|}{|L| + |R| + |S|} \cdot \frac{|L| + |R| - 1}{|L| + |R| + |S| - 1} \cdots \frac{1}{|S| + 1} = \frac{(|L| + |R|)! \cdot |S|!}{(|L| + |R| + |S|)!}$$

and therefore,

$$Pr(TC_2) = \frac{|M|}{N} \cdot \frac{(|L| + |R|)! \cdot |S|!}{(|L| + |R| + |S|)!}$$

Case of TC_1: For a transfer charge of 1 to occur, it must be that an interval from M arrives first, and then one of two cases: all intervals from L (respectively R) arrive, followed by the first interval of S, and the last interval of R (respectively L) arrives after. These two cases are symmetrical and we'll focus on the first one, which can be visualized as follows:

$$\text{first } M \rightarrow \text{last } L \rightarrow \text{first } S \rightarrow \text{last } R$$

Consider the following two events:
Event A_L: The first interval from S arrives after the last interval from L.
Event B_R: The last interval from R arrives after the first interval from S.

We want to compute $Pr(A_L \cap B_R)$. Notice that in the previous case of TC_2 we computed $Pr(A_L \cap \overline{B_R})$. We get that:

$$Pr(\overline{B_R}|A_L) = \frac{Pr(A_L \cap \overline{B_R})}{Pr(A_L)} = \frac{(|L| + |R|)! \cdot (|L| + |S|)!}{(|L| + |R| + |S|)! \cdot |L|!}$$

$$Pr(A_L \cap B_R) = Pr(B_R|A_L) \cdot Pr(A_L) = (1 - Pr(\overline{B_R}|A_L)) \cdot Pr(A_L)$$

$$= \left(1 - \frac{(|L| + |R|)! \cdot (|L| + |S|)!}{(|L| + |R| + |S|)! \cdot |L|!}\right) \cdot \frac{|L|! \cdot |S|!}{(|L| + |S|)!}$$

$$= \frac{|L|! \cdot |S|!}{(|L| + |S|)!} - \frac{(|L| + |R|)! \cdot |S|!}{(|L| + |R| + |S|)!}$$

Similarly, for the symmetrical case we get that:

$$Pr(A_R \cap B_L) = \frac{|R|! \cdot |S|!}{(|R| + |S|)!} - \frac{(|L| + |R|)! \cdot |S|!}{(|L| + |R| + |S|)!}$$

Combining the two cases we get that:

$$Pr(TC_1) = \frac{M}{N} \left[Pr(A_L \cap B_R) + Pr(A_R \cap B_L) \right]$$

$$= \frac{M}{N} \left[\frac{|L|! \cdot |S|!}{(|L| + |S|)!} + \frac{|R|! \cdot |S|!}{(|R| + |S|)!} - 2\frac{(|L| + |R|)! \cdot |S|!}{(|L| + |R| + |S|)!} \right]$$

Finally, we have that:

$$\mathbb{E}[TC(I)] = Pr(TC_1) + 2Pr(TC_2) = \frac{M}{N} \left[\frac{|L|! \cdot |S|!}{(|L| + |S|)!} + \frac{|R|! \cdot |S|!}{(|R| + |S|)!} \right]$$

We have that $(|L|, |R|, |S|, |M|) \in \{\mathbb{N}^*\}^4$. We will also assume that $|S| \geq 3$. We deal with the cases of $|S| = 1$ and $|S| = 2$ in appendix A. To maximize $\mathbb{E}[TC(I)]$ we set $|L| = |R| = 1$, and $|S| = 3$, and we get:

$$\mathbb{E}[TC(I)]) \leq \frac{|M|}{|M| + 5} \cdot \frac{2 \cdot 3!}{4!} = \frac{1}{2} \frac{|M|}{|M| + 5} \xrightarrow{|M| \to +\infty} \frac{1}{2}$$

Therefore, $\mathbb{E}[\Phi(I)] = \mathbb{E}[TC(I) + DC(I)] \leq \frac{1}{2} + 2$. □

Corollary 3.2. *Algorithm 1 is* $2.5-$*competitive on instances of the form as in figure 3.*

We are now ready to prove the following Theorem for the general case of $k = 2$.

Theorem 3.3. *Algorithm 1 achieves a competitive ratio of* 2.5 *for the problem of interval selection on instances with at most two different lengths.*

Proof. Let $C_i = (L_i, R_i, M_i, S_i)$ denote a basic construction (or sub-instance) that follows the structure described earlier, with $L_i \cup R_i \neq \emptyset$. Given an optimal solution OPT, any instance can be partitioned into a set of such constructions, each being uniquely identified by its middle non-optimal intervals. Let $\mathcal{C} = \{C_1, C_2, ..., C_n\}$ denote the set of all these constructions. Although it could be that $|M_i| \gg 1$, we will abuse the notation and refer to the *interval* M_i. Figure 4 shows an instance that is partitioned into three basic constructions: $C_1 = (\emptyset, \{I_1\}, \{M_1\}, \{J_1, J_2, J_3\})$, $C_2 = (\{I_1\}, \{I_3\}, \{M_2\}, \{I_2, J_4\})$, and $C_3 = (\{I_2\}, \{I_4\}, \{M_3\}, \{J_4, I_3\})$.

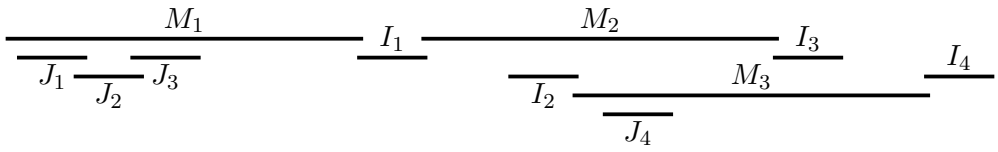


Figure 4: Example instance with three basic constructions.

We focus on these constructions, because a transfer charge can only occur to the intervals in $(S_1 \cup S_2 \cup ... \cup S_n)$. It is helpful to associate the event of a transfer charge with the related construction, and note that no transfer charge will be associated with that construction again. For example, after interval J_4 is taken (fig. 4), no transfer charge can be associated with C_2 and C_3. This is because whenever an interval from S_i is accepted, it can never be replaced again, and M_i cannot be accepted again. Consider the middle intervals $\{M_1, ..., M_n\}$. A subset of those intervals will be taken by the algorithm during the execution. Whenever M_i is taken, a transfer charge may occur. Intervals outside C_i can affect it by blocking intervals in $L_i \cup R_i$ (before or after M_i is taken), or S_i (before M_i is taken). In every case, the expected total amount of charge on any S_i is no more compared to having C_i separately, which, in addition to the fact that any interval is transfer charged at most once, gives us the desired result.

□

3.2 Case of $k > 2$

In the case of $k > 2$, the nesting depth can be greater than 1, which allows for a predecessor trace of length greater than 2. As before, we fix an optimal solution and consider the set of all basic constructions $\mathcal{C}$ derived from the instance. We note that a basic construction can now be contained in another. More specifically, if C_i is contained in C_j, it means that $(L_i \cup R_i \cup M_i \cup S_i) \subseteq S_j$. For every interval $J \in ALG$, we consider the predecessor trace $P_J = (M_1, ..., M_d, J)$ that was formed during the execution. When an interval (M_i) is replaced by another (M_{i+1}), it also transfers all of its charge, and we have that $\Phi(M_i) \leq \Phi(M_{i+1}) \leq \Phi(M_i) + 2$. W.l.o.g. we assume that $M_1, ..., M_d$ are all middle intervals of some basic constructions. The only way this isn't true is if for some i, M_i does not partially conflict with any optimal intervals, in which case it cannot increase the amount of charge transferred to M_{i+1}. Interval J may or may not correspond to a middle interval, but we know that $J \in S_d$. The total amount of charge on any one interval is a random quantity, and we want to upper bound $\mathbb{E}[\Phi(J) \mid J \in ALG]$ for every predecessor trace P_J.

Lemma 3.4. $\mathbb{E}[\Phi(J) \mid J \in ALG] \leq 2.5$ *for every* $J \in ALG$ *and predecessor trace* P_J.

Proof. As before, we have that $DC(J) \leq 2$ and we focus on $\mathbb{E}[TC(J)]$. Notice that for every P_J, the expected amount of charge added to J because of interval M_i depends on the subset of intervals of $(L_i \cup R_i \cup S_i)$ that are yet to arrive after M_i was accepted. We are able to derive a bound on $\mathbb{E}[TC(J)]$ by assuming the last interval of L_i and the last interval of R_i are yet to arrive, and lower bounding the remaining of S_i by the fact that intervals $M_{i+1}, ..., J$ are yet to arrive.

Let $(S_i', L_i', R_i') \subseteq (S_i, L_i, R_i)$ be the set of intervals pending to arrive after M_i was accepted, and for $i > 1$, let $M_i' \subseteq M_i$ be the set of intervals pending to arrive after M_{i-1} was accepted. Similarly, let $M_1' \subseteq M_1$ be the set of intervals pending after M_1 was able to be accepted by the

algorithm. For readability, we omit $|\cdot|$ and refer to $|M'_i|, |S'_i|, |L'_i|$, and $|R'_i|$ as M'_i, S'_i, L'_i, R'_i. Let $N'_i = M'_i + L'_i + R'_i + S'_i$. For $1 \leq i \leq d$ we have that:

$$S'_i \geq S'_d + \sum_{j=i+1}^{d} (M'_j + L'_j + R'_j) \tag{1}$$

We first consider the case of $S'_d \geq 3$. From the analysis of Lemma 3.1 we get that:

$$\mathbb{E}[TC(J)] = \frac{M'_1}{N'_1} \cdot \left[\frac{L'_1! \cdot S'_1!}{(L'_1 + S'_1)!} + \frac{R'_1! \cdot S'_1!}{(R'_1 + S'_1)!}\right] + \cdots + \frac{M'_d}{N'_d} \cdot \left[\frac{'L_d! \cdot S'_d!}{(L'_d + S'_d)!} + \frac{R'_d! \cdot S'_d!}{(R'_d + S'_d)!}\right]$$

$$\leq \frac{M'_1}{M'_1 + 2 + S'_1} \cdot \left[2\frac{S'_1!}{(S'_1 + 1)!}\right] + \cdots + \frac{M'_d}{M'_d + 2 + S'_d} \left[2\frac{S'_d!}{(S'_d + 1)!}\right]$$

$$= 2\left[\frac{M'_1}{(M'_1 + 2 + S'_1) \cdot (S'_1 + 1)} + \cdots + \frac{M'_d}{(M'_d + 2 + S'_d) \cdot (S'_d + 1)}\right]$$

$$\leq 2\left[\frac{M'_1}{5 + 2(d-1) + \sum_{j=1}^{d} M'_j} \cdot \frac{1}{4 + 2(d-1) + \sum_{j=2}^{d} M'_j} + \cdots + \frac{M'_d}{M'_d + 5} \cdot \frac{1}{4}\right]$$

$$= 2\sum_{i=1}^{d} \frac{M'_i}{(5 + 2(d-i) + \sum_{j=i}^{d} M'_j) \cdot (4 + 2(d-i) + \sum_{j=i+1}^{d} M'_j)}$$

The first inequality is because we set $L'_i = R'_i = 1$. The second inequality is because of (1).

Let $F^d(x_1, .., x_d) = \sum_{i=1}^{d} \frac{x_i}{(5+2(d-i)+\sum_{j=i}^{d} x_j) \cdot (4+2(d-i)+\sum_{j=i+1}^{d} x_j)}$, with $x_i \geq 1$ for every i. For readability, we will write $F^d(x_1, ..., x_d) = \sum_{i=1}^{d} \frac{x_i}{(x_i+2+s_i) \cdot (s_i+1)}$, with $s_i = x_{i+1} + 2 + s_{i+1}$, and $s_d = 3$. We will show that for any d and $x_1, ..., x_d$:

$$F^d(x_1, ..., x_d) \leq \frac{1}{4}$$

We show this by induction on d:

Base case $d = 1$: $F^1(x_1) = \frac{x_1}{4(x_1+5)} \leq \frac{1}{4}$ holds.

Induction step: For $d = D$, we assume that $F^D(x_1, .., x_D) \leq \frac{1}{4}$.

For $d = D + 1$, we focus on the first two terms of the sum:

$$\sum_{i=1}^{2} \frac{x_i}{(x_i + 2 + s_i) \cdot (s_i + 1)} = \frac{x_1}{(x_1 + 2 + s_1) \cdot (s_1 + 1)} + \frac{x_2}{(x_2 + 2 + s_2) \cdot (s_2 + 1)}$$

$$= \frac{x_1}{(x_1 + x_2 + 4 + s_2) \cdot (x_2 + 3 + s_2)} + \frac{x_2}{(x_2 + 2 + s_2) \cdot (s_2 + 1)}$$

We will show that:

$$\sum_{i=1}^{2} \frac{x_i}{(x_i + 2 + s_i) \cdot (s_i + 1)} \leq \frac{x_1 + x_2}{(x_1 + x_2 + 2 + s_2) \cdot (s_2 + 1)} \tag{2}$$

We have that:

$$\frac{x_1 + x_2}{(x_1 + x_2 + 2 + s_2) \cdot (s_2 + 1)} - \frac{x_1}{(x_1 + x_2 + 4 + s_2) \cdot (x_2 + 3 + s_2)} - \frac{x_2}{(x_2 + 2 + s_2) \cdot (s_2 + 1)} \geq$$

$$\frac{x_1}{(x_1 + x_2 + 2 + s_2) \cdot (s_2 + 1)} - \frac{x_1}{(x_1 + x_2 + 4 + s_2) \cdot (x_2 + 3 + s_2)} \geq 0$$

Therefore (2) holds, and we have that $F^{D+1}(x_1, x_2, .., x_{D+1}) \leq F^D(x_1 + x_2, x_3, .., x_{D+1})$, which we know is at most $\frac{1}{4}$ by the induction hypothesis.

Putting everything together, we have that $\mathbb{E}[TC(J)] \leq \frac{1}{2}$, and because $DC(J) \leq 2$, we get that $\mathbb{E}[\Phi(J)] \leq 2.5$. The cases of $s_d = 1$ and $s_d = 2$ are dealt with in appendix A. □

Corollary 3.5. *Algorithm 1 is* 2.5*-competitive for the problem of interval selection for all* k*.*

4 Randomness Extraction

Our analysis of the competitiveness for unweighted interval selection is another example of the power of random-order arrivals (vs adversarial order) for deterministic algorithms. A basic question in online algorithms is the power of random-order deterministic algorithms relative to adversarial order randomized algorithms. We know that there are problems where deterministic random-order algorithms provide provably better competitive ratios than randomized algorithms (e.g. the secretary problem [22]). But are there problems where adversarial order randomized algorithms are provably (or even seemingly) better than random-order deterministic algorithms? It is natural then to see if we can use the randomness in the arrival of input items to extract random bits. Such bits may be used to derandomize certain algorithms in the random-order model. *Barely random* algorithms [25] use a (small) constant number of random bits, and are well suited to be considered for this purpose. These algorithms are often used in the *classify and randomly select* paradigm, where inputs are partitioned into a small number of classes, and the algorithm randomly selects a class of items to work with. We consider the simple 1-bit randomness extraction process as described in Process 2. We assume there exist at least two different classes, or item types, that all input items belong to. Furthermore, we maintain a counter that represents the number of items that have arrived so far. Our process returns 1, if the first item of the second type to arrive online is on an even counter. The adversary can choose the number of items of each type to affect the probability of getting either output, but we show that the worst case bias of this bit is at most $\frac{2}{3}$.

Process 2 Biased bit extraction

On the arrival of I_i:
if $i = 1$ **then**
  $type \leftarrow type(I_1)$
else
  if $type \neq type(I_i)$ **then**
    return$(1 - (i \mod 2))$ and terminate the process

Theorem 4.1. *The bit extracted by process 2 has a worst case bias of at most* $\frac{2}{3}$.

Proof. Let there be A items of $type_A$ and B items of $type_B$, with $N = A + B$. Let E_v be the event where the second item type arrives on an even counter. Let also A_e (respectively B_e) be the event that the first appearance of $type_A$ ($type_B$) is on an even counter, and F_A (resp. F_B) be the event that the very first item that arrives is of $type_A$. We assume that N is very large, and we are sampling from an infinite population. We have that:

$$Pr(E_v) = Pr(B_e|F_A) \cdot Pr(F_A) + Pr(A_e|F_B) \cdot Pr(F_B)$$

We start by computing $Pr(B_e|F_A)$:

$$Pr(B_e|F_A) = \frac{B}{N} + \left(\frac{A}{N}\right)^2 \cdot \frac{B}{N} + \left(\frac{A}{N}\right)^4 \cdot \frac{B}{N} + \left(\frac{A}{N}\right)^6 \cdot \frac{B}{N} + \cdots = \frac{B}{N} \sum_{i=0}^{+\infty} \left(\frac{A}{N}\right)^{2i}$$

and therefore:

$$Pr(B_e|F_A) \cdot Pr(F_A) = \frac{AB}{N^2} \sum_{i=0}^{+\infty} \left(\frac{A}{N}\right)^{2i}$$

Similarly, we get that:

$$Pr(A_e|F_B) \cdot Pr(F_B) = \frac{AB}{N^2} \sum_{i=0}^{+\infty} \left(\frac{B}{N}\right)^{2i}$$

Putting everything together:

$$Pr(E_v) = \frac{AB}{N^2} \sum_{i=0}^{+\infty} \left(\frac{A^{2i} + B^{2i}}{N^{2i}}\right)$$

Let $A = \alpha N$ with $\alpha \in (0, 1)$. We can rewrite $Pr(E_v)$ as follows:

$$Pr(E_v) = \frac{\alpha(1-\alpha)N^2}{N^2} \sum_{i=0}^{+\infty} \frac{(\alpha N)^{2i} + (1-\alpha)^{2i} N^{2i}}{N^2}$$

$$= \alpha(1-\alpha) \sum_{i=0}^{+\infty} \alpha^{2i} + (1-\alpha)^{2i}$$

$$= \alpha(1-\alpha) \frac{-2\alpha^2 + 2\alpha + 1}{(\alpha - 2)\alpha(\alpha^2 - 1)}$$

$$= \frac{2\alpha^2 - 2\alpha - 1}{(\alpha + 1)(\alpha - 2)}$$

Let $f(\alpha) = \frac{2\alpha^2-2\alpha-1}{(\alpha+1)(\alpha-2)}$. We have that $f[(0,1)] \in (\frac{1}{2}, \frac{2}{3})$ (figure 5). In conclusion, the worst case bias of the bit extracted through Process 2 is $\frac{2}{3}$. □

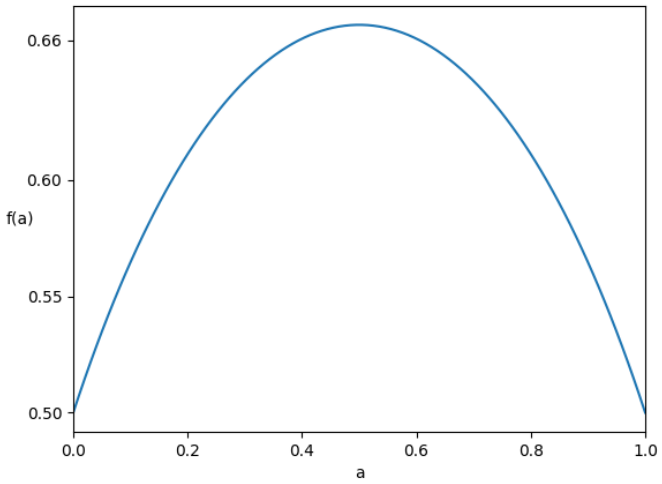


Figure 5: Plot of $f(a)$

Fung et al. [10] give barely random algorithms for some weighted variations of interval selection in the real-time model. Their 2-competitive algorithm for the case of single-length intervals with arbitrary weights is also directly applicable to the any-order model, and consequently to the random-order model, while maintaining the same competitiveness. The real line is divided into slots, and each interval can be viewed as belonging to an *even* or *odd* slot. Their algorithm uses one random bit to pick one slot type, and gets an optimal solution amongst those intervals. We refer to [10] for a complete description of the algorithm. We can derandomize this algorithm using a random bit extracted by Process 2 as follows: Our algorithm starts working on the first type of intervals that arrive as if it was the chosen one. When a new (slot-)type arrives, our bit is extracted, and we decide whether we will switch to the second type intervals or not. With a slight change in their proof (Theorem 3.1 in [10]) we see that our algorithm is 3-competitive.

We note that revoking is essential in the above algorithm. Although the 2-competitive algorithm by Fung et al. already requires that revoking is allowed in the model, we also need to be able to discard the entire solution constructed by the time the random bit is extracted. Process 2 may be used more generally to derandomize algorithms that fall under the classify and randomly select paradigm, when two classes are used. Consider a deterministic algorithm A, and let ALG_1 (resp. ALG_2) denote the performance of the algorithm on input items that belong to class 1 (resp. 2). Assuming $ALG_1 + ALG_2 \geq \frac{OPT}{c}$, with $c \geq 1$, we get a $3c$-competitive algorithm.

Under the assumption that there exists a global ordering amongst all online items, we are able to extract an unbiased bit simply by comparing the two first items to arrive (Process 3).

Theorem 4.2. *Under the assumption that there exista a global ordering amongst all items in the input instance, process 3 (Distinct-Unbiased) produces an unbiased bit.*

Process 3 Distinct-Unbiased

On the arrival of I_1, I_2:
if $I_1 < I_2$ **then**
    return(1)
else
    return(0)

Proof. Let $I_1, I_2, ..., I_N$ be all the items in the instance, such that $I_1 < I_2 < ... < I_N$. Let I_a, I_b be the first two items that arrive. Let E_1 denote the event that $I_a < I_b$. Let F_i denote the event that item I_i arrives first, and S_i denote the event that $I_b > I_i$. We have that:

$$Pr(E_1) = Pr(S_1|F_1) \cdot Pr(F_1) + \cdots + Pr(S_N|F_N) \cdot Pr(F_N)$$

$$= \frac{1}{N}\frac{N-1}{N-1} + \cdots + \frac{1}{N}\frac{N-N}{N-1}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\frac{N-i}{N-1} = \frac{1}{2}$$

□

We can repeat this process for the next pair of online items. Generally, given $2N$ online items we can extract N unbiased bits. For example, this process could be applicable to interval scheduling under the assumption that all intervals have distinct starting times. In this setting, we can use Process 3 to derandomize the algorithm by Fung et al. for single-length arbitrary weights, and maintain its competitiveness. We can even combine these two processes, and get a 6-competitive deterministic algorithm for the case of arbitrary weights and two different lengths. The algorithm would use the unbiased bit from the first two intervals to decide on the length. While working on any length, we would use Process 2 to decide on the slot type.

5 Conclusions

We have shown an upper bound of 2.5 on Algorithm 1 under random arrivals. We have also given a lower bound of 2 (as $n \to +\infty$) for that algorithm. We believe a matching upper bound can be shown with a more careful analysis of direct charging. It is also plausible that a deterministic algorithm can be better than 2-competitive. A better algorithm might have additional replacement rules, in particular for partial conflicts. We also want to improve the $\frac{12}{11}$ lower bound for random arrivals assuming that this is not the optimal bound. Our study has this far only considered deterministic algorithms and an obvious question is to consider randomized algorithms for interval selection with revoking in the random order model.

We studied two processes for extracting random bits from uniformly random arrivals. This may be applied to other problems where global distinctions among items can be made. We think it would be interesting to consider models with a large number of items arriving in between random choices. This may allow sufficiently many bits to be extracted in order to simulate the next random choice, and derandomize *classify and randomly select* algorithms with many classes of items.

References

[1] Unnar Th Bachmann, Magnús M Halldórsson, and Hadas Shachnai. "Online selection of intervals and t-intervals". In: *Information and Computation* 233 (2013), pp. 1–11.

[2] Allan Borodin and Christodoulos Karavasilis. "Any-order online interval selection". In: *International Workshop on Approximation and Online Algorithms*. Springer. 2023, pp. 175–189.

[3] Joan Boyar, Lene M Favrholdt, Michal Kotrbčík, and Kim S Larsen. "Relaxing the irrevocability requirement for online graph algorithms". In: *Algorithmica* 84.7 (2022), pp. 1916–1951.

[4] Niv Buchbinder, Moran Feldman, and Roy Schwartz. "Online submodular maximization with preemption". In: *ACM Transactions on Algorithms (TALG)* 15.3 (2019), pp. 1–31.

[5] Ran Canetti and Sandy Irani. "Bounding the power of preemption in randomized scheduling". In: *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*. 1995, pp. 606–615.

[6] Yuval Emek, Magnús M Halldórsson, and Adi Rosén. "Space-constrained interval selection". In: *ACM Transactions on Algorithms (TALG)* 12.4 (2016), pp. 1–32.

[7] Leah Epstein and Asaf Levin. "Improved randomized results for that interval selection problem". In: *European Symposium on Algorithms*. Springer. 2008, pp. 381–392.

[8] Ulrich Faigle and Willem M Nawijn. "Note on scheduling intervals on-line". In: *Discrete Applied Mathematics* 58.1 (1995), pp. 13–17.

[9] Jon Feldman, Nitish Korula, Vahab Mirrokni, Shanmugavelayutham Muthukrishnan, and Martin Pál. "Online ad assignment with free disposal". In: *International workshop on internet and network economics*. Springer. 2009, pp. 374–385.

[10] Stanley PY Fung, Chung Keung Poon, and Feifeng Zheng. "Improved randomized online scheduling of intervals and jobs". In: *Theory of Computing Systems* 55.1 (2014), pp. 202–228.

[11] Virginie Gabrel. "Scheduling jobs within time windows on identical parallel machines: New model and algorithms". In: *European Journal of Operational Research* 83.2 (1995), pp. 320–329.

[12] Juan A Garay, Inder S Gopal, Shay Kutten, Yishay Mansour, and Moti Yung. "Efficient on-line call control algorithms". In: *Journal of Algorithms* 23.1 (1997), pp. 180–194.

[13] Mohit Garg, Debajyoti Kar, and Arindam Khan. "Random-Order Online Interval Scheduling and Geometric Generalizations". In: *arXiv preprint arXiv:2402.14201* (2024).

[14] Anupam Gupta and Sahil Singla. "Random-Order Models". In: *Beyond the Worst-Case Analysis of Algorithms*. Ed. by Tim Roughgarden. Cambridge University Press, 2020, pp. 234–258. DOI: 10.1017/9781108637435.015. URL: https://doi.org/10.1017/9781108637435.015.

[15] Gupta, Lee, and Leung. "An optimal solution for the channel-assignment problem". In: *IEEE Transactions on Computers* 100.11 (1979), pp. 807–810.

[16] Nicholas G Hall and Michael J Magazine. "Maximizing the value of a space mission". In: *European journal of operational research* 78.2 (1994), pp. 224–241.

[17] Sungjin Im and Yajun Wang. "Secretary problems: Laminar matroid and interval scheduling". In: *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*. SIAM. 2011, pp. 1265–1274.

[18] Kazuo Iwama and Shiro Taketomi. "Removable online knapsack problems". In: *Automata, Languages and Programming: 29th International Colloquium, ICALP 2002 Málaga, Spain, July 8–13, 2002 Proceedings 29*. Springer. 2002, pp. 293–305.

[19] Antoon WJ Kolen, Jan Karel Lenstra, Christos H Papadimitriou, and Frits CR Spieksma. "Interval scheduling: A survey". In: *Naval Research Logistics (NRL)* 54.5 (2007), pp. 530–543.

[20] Mikhail Y Kovalyov, Chi To Ng, and TC Edwin Cheng. "Fixed interval scheduling: Models, applications, computational complexity and algorithms". In: *European journal of operational research* 178.2 (2007), pp. 331–342.

[21] Richard J Lipton and Andrew Tomkins. "Online Interval Scheduling." In: *SODA*. Vol. 94. 1994, pp. 302–311.

[22] Aranyak Mehta. "Online Matching and Ad Allocation". In: *Foundations and Trends® in Theoretical Computer Science* 8.4 (2013), pp. 265–368.

[23] Serge Plotkin. "Competitive routing of virtual circuits in ATM networks". In: *IEEE Journal on Selected Areas in Communications* 13.6 (1995), pp. 1128–1136.

[24] Dror Rawitz and Adi Rosén. "Online budgeted maximum coverage". In: *Algorithmica* 83 (2021), pp. 2989–3014.

[25] Nick Reingold, Jeffery Westbrook, and Daniel D Sleator. "Randomized competitive algorithms for the list update problem". In: *Algorithmica* 11.1 (1994), pp. 15–32.

[26] Steven S Seiden. "Randomized online interval scheduling". In: *Operations Research Letters* 22.4-5 (1998), pp. 171–177.

[27] Gerhard J Woeginger. "On-line scheduling of jobs with fixed start and end times". In: *Theoretical Computer Science* 130.1 (1994), pp. 5–16.

[28] Neal E Young. "K-medians, facility location, and the Chernoff-Wald bound". In: *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*. 2000, pp. 86–95.

A Dealing with the cases of $|S| < 3$

A.1 In Lemma 3.1.

Consider the case of $|S| = 1$, with $S = \{S_1\}$. As per the analysis of Lemma 3.1, we get that $\mathbb{E}[TC(S_1)] \leq 1$. If S_1 is an optimal interval, it means that it is only directly charged when it is accepted, and we have that $DC(S_1) = 1$. Therefore, $\mathbb{E}[\Phi(S_1)] \leq 2$. If S_1 is not an optimal interval (because it conflicts with the left and/or right optimal interval), we have that $\Phi(S_1) \leq 2$ just from the fact that there exist two optimal intervals in total.

We now consider the case of $|S| = 2$, with $S = \{S_1, S_2\}$. We have that for $I \in \{S_1, S_2\}$, $\mathbb{E}[TC(I)] \leq \frac{2}{3}$. If both S_1, S_2 are optimal intervals, we have that $DC(S_1) = DC(S_2) = 1$, and therefore for $I \in \{S_1, S_2\}$, $\mathbb{E}[\Phi(I)] \leq \frac{2}{3} + 1 < 2.5$. If S_1, S_2 are both non-optimal intervals, we have that $\Phi(S_i) \leq 2$ for all i, because there exist at most two optimal intervals in total. Consider now the case of S_1 being an optimal interval, and S_2 being a non-optimal interval. We consider two subsequent cases:

Case 1: S_1 and S_2 don't overlap. In this case we have that $DC(S_1) = 1$, and $DC(S_2) \leq 1$, since S_2 must be in conflict with either the left or the right optimal interval. Therefore we get $\mathbb{E}[\Phi(I)] < 2.5$ for $I \in \{S_1, S_2\}$.

Case 2: S_1 and S_2 overlap. In this case, we have that $DC(S_1) = 1$, and $1 \leq DC(S_2) \leq 2$. Let $I \in \{S_1, S_2\}$ be the interval that makes it into the final solution. We have that $\mathbb{E}[DC(I)] \leq \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 2 = 1.5$, and $\mathbb{E}[\Phi(I)] \leq \frac{2}{3} + 1.5 < 2.5$.

In conclusion, Lemma 3.1 holds even in the case of $|S| < 3$.

A.2 In Lemma 3.4.

The induction argument in Lemma 3.4 goes through with $|S'_d| = 1$ and $|S'_d| = 2$, giving us a bound on the expected amount of transfer charge of 1 and $\frac{2}{3}$ respectively. As in A.1, in the case of $|S'_d| = 1$ with S_1 being an optimal interval, we have that $DC(S_1) = 1$. If S_1 is not an optimal interval and conflicts with one optimal interval, we have that $DC(S_1) \leq 1$. If S_1 conflicts with two optimal intervals, we have that $DC(S_1) = 2$ with probability at most $\frac{1}{3}$. Therefore $\mathbb{E}[DC(S_1)] \leq \frac{2}{3} + 2 \cdot \frac{1}{3} = \frac{4}{3}$. In all cases, we have that $\mathbb{E}[TC(S_1)] + \mathbb{E}[DC(S_1)] \leq 2.5$.

In the case of $S'_d = \{S_1, S_2\}$ with both intervals being optimal, as in A.1, we get that $DC(S_1) = DC(S_2) = 1$. If neither interval is optimal, similar to the argument in the case of $|S'_d| = 1$, we have that $DC(S_i) = 2$ with probability at most $\frac{1}{2}$, and $\mathbb{E}[DC(S_i)] \leq \frac{1}{2} + 2 \cdot \frac{1}{2}$. Therefore, $\mathbb{E}[\Phi(S_i)] \leq 2.5$. Finally, the case of one of the two intervals being optimal is handled like in A.1.

B A lower bound under random arrivals.

We will show a lower bound of $\frac{12}{11}$ for all deterministic algorithms with revoking in the random-order model. This is in contrast to the real-time model with revoking, where 1-competitiveness is attainable. Let LB_1 be a three interval instance as shown in figure 6.

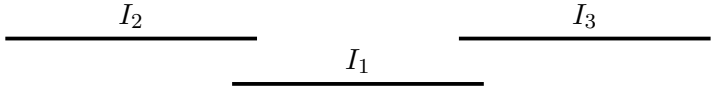


Figure 6: Instance LB_1.

First, notice that because the algorithm has no knowledge of the size of the input, it must act greedily on the first interval to arrive. If that weren't the case, we could introduce a one-interval instance where the competitiveness of the algorithm would be unbounded. Consider now the behaviour of the algorithm if I_1 was to arrive first.

Case 1: The algorithm will not replace I_1 with either interval that might arrive second. In this case, we know that with probability at least $\frac{1}{3}$, the algorithm will have one interval in its solution.

Case 2: There is at least one interval in $\{I_2, I_3\}$ such that if it is the second interval to arrive, it will replace I_1. Let I_3 be such an interval. We can then use instance LB_2 (fig. 7), with intervals I_1 and I_3 being the same as in LB_1. In this case we know that with probability at least $\frac{1}{6}$, the algorithm will have one interval in its solution.

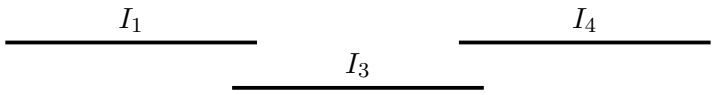


Figure 7: Instance LB_2.

In conclusion, there is always an instance with $OPT = 2$, and $\mathbb{E}[ALG] \leq \frac{1}{6} + 2 \cdot \frac{5}{6} = \frac{11}{6}$, and therefore the competitive ratio is at least $\frac{12}{11}$. Even under the assumption that the algorithm knows the size of the input, the same bound holds. If I_1 was not taken upon arrival, using instance LB_2 we know that the algorithm will only have one interval in its solution with probability at least $\frac{1}{3}$.