Any-Order Online Interval Selection

Allan Borodin and Christodoulos Karavasilis(✉)

University of Toronto, Toronto, Canada
{bor,ckar}@cs.toronto.edu

Abstract. We consider the problem of online interval scheduling on a single machine, where intervals arrive online in an order chosen by an adversary, and the algorithm must output a set of non-conflicting intervals. Traditionally in scheduling theory, it is assumed that intervals arrive in order of increasing start times. We drop that assumption and allow for intervals to arrive in any possible order. We call this variant *any-order interval selection* (AOIS). We assume that some online acceptances can be revoked, but a feasible solution must always be maintained. For unweighted intervals and deterministic algorithms, this problem is unbounded. Under the assumption that there are at most k different interval lengths, we give a simple algorithm that achieves a competitive ratio of $2k$ and show that it is optimal amongst deterministic algorithms, and a restricted class of randomized algorithms we call *memoryless*, contributing to an open question by Adler and Azar [1]; namely whether a randomized algorithm without memory or with only "bounded" access to history can achieve a constant competitive ratio. We connect our model to the problem of *call control* on the line, and show how the algorithms of Garay et al. [22] can be applied to our setting, resulting in an optimal algorithm for the case of proportional weights. We also discuss the case of intervals with arbitrary weights, and show how to convert the single-length algorithm of Fung et al. [20] into a *classify and randomly select* algorithm that achieves a competitive ratio of $2k$. Finally, we consider the case of intervals arriving in a *random order*, and show that for single-lengthed instances, a *one-directional* algorithm (i.e. replacing intervals in one direction), is the only deterministic memoryless algorithm that can possibly achieve a strict competitive ratio less than 2.

Keywords: interval selection · scheduling · online algorithms · call control

1 Introduction

We consider the problem of scheduling intervals online with revoking[1]. Intervals arrive with a fixed start time and fixed end time, and have to be taken right away, or be discarded upon arrival, while no intervals in the solution conflict.

[1] Displacing one or more previously scheduled intervals with a conflicting new interval.

J. Byrka and A. Wiese (Eds.): WAOA 2023, LNCS 14297, pp. 175–189, 2023.
https://doi.org/10.1007/978-3-031-49815-2_13

The algorithm has to decide which intervals to include in the final schedule, so as to optimize some objective.

In the unweighted case, the goal is to maximize the number of intervals in the final solution. In the weighted case, we want an interval-set of maximum weight. Following previous work, we allow some revoking of online decisions, which is often considered even in the conventional start-time-ordered scheduling model. More precisely, if a newly arrived interval conflicts with other intervals already taken by the algorithm, we are able to take the new interval and discard the conflicting intervals. We are able to displace multiple existing intervals at once, although this won't be needed in the unweighted case. To avoid confusion, we should note that *preemption*[2] is often used in the interval selection literature to mean precisely this revoking of previous decisions we just described. Under this definition, preemption is allowed in our model. When we discard an interval it is final and it cannot be taken again.

We focus mainly on the unweighted case, where all intervals have the same weight. We discuss the competitive ratio of the problem in terms of k, the number of distinct interval lengths. However our algorithm does not need a priori knowledge of k. We show that a simple, deterministic, "memoryless" algorithm that only replaces when the new interval is entirely subsumed by an existing one, achieves the optimal competitive ratio in terms of the parameter k. We also show that "memoryless" randomized algorithms can not do any better. The main difference between our model and most of the interval selection literature, is allowing intervals to arrive in any order, a strict generalization of the ordered case. Bachmann et al. [5] have studied the any-order input model in the context of "t-intervals" (we are concerned with $t = 1$). They consider randomized algorithms, and don't allow revoking. In that model, they get a lower bound of $\Omega(N)$, with N being the number of intervals in a given input instance. The next most closely related problem is that of call admission [21] on the line graph, with online intervals corresponding to paths of a given line graph. The connection between call control on the line graph and interval selection has been noted before, but has not been carefully defined. We wish to clarify this connection by explaining the similarities as well as the differences, and how results correspond. We note that the parameter $k \leq N$ (respectively, $k \leq n - 1$) is an obvious refinement of the number of intervals (respectively, the number of vertices for call admission on a line graph with n vertices).

The applications of interval selection problems are plentiful. Some examples are resource allocation, network routing, transportation, and computer wiring. We refer the reader to the surveys by Kolen et al. [27], and Kovalyov et al. [30] for an overview of results and applications in the area of interval scheduling.

Related Work. Lipton and Tomkins [31] introduced the online interval scheduling problem. In our terminology, they consider the arrival of intervals with increasing start times (ordered), and interval weights that are proportional to

[2] In contrast to revoking, preemption in much of the scheduling literature means the pausing of a scheduled job, and resuming it later.

the lengths. They don't allow displacement of existing intervals, and give a randomized algorithm with competitive ratio $O((log\Delta)^{1+\epsilon})$, where Δ is the ratio of the longest to shortest interval.

In the unweighted case with increasing starting times, Faigle and Nawijn [17] give an optimal 1-competitive algorithm that is allowed to revoke previous decisions (replace intervals). In the weighted case with increasing starting times, Woeginger [37] shows that for general weights, no deterministic algorithm can achieve a constant competitive ratio. Canetti and Irani [11] extend this and show that even randomized algorithms with revocable decisions cannot achieve a constant ratio for the general weighted case. For special classes of weight functions based on the length (including proportional weights), Woeginger [37] gives an optimal deterministic algorithm with competitive ratio 4. Seiden [34] gives a randomized $(2+\sqrt{3})$-competitive algorithm when the weight of an interval is given by a continuous convex function of the length. Epstein and Levin [16] give a 2.45-competitive randomized algorithm for weights given by functions of the length that are monotonically decreasing, and they also give an improved $1 + \ln(2) \approx 1.693$ upper bound for the weight functions studied by Woeginger [37]. Fung et al. [20] provide the best known upper bounds, giving *barely random* algorithms that achieve a competitive ratio of 2 for all the Woeginger weight functions. These algorithms randomly choose one of two deterministic algorithms at the beginning. More generally, barely random algorithms have access to a small number of deterministic algorithms, and randomly choose one.

Restricting interval lengths has previously been considered in the literature, e.g. Lipton and Tomkins [31] study the case of two possible lengths, and Bachmann et al. [5] consider single and two-length instances. For the related offline problem of throughput maximization, Hyatt-Denesik et al. [25] consider c distinct processing times. The special case of single-length jobs has been studied in the job scheduling [6,13,35], sum coloring [9], and the interval selection literature [19,32]. Woeginger [37] also points out how his results can be extended to the case of equal lengths and arbitrary weights. Miyazawa and Erlebach [32] point out the equivalency between fixed length (w.l.o.g. unit) instances, and proper interval instances, i.e. instances where no interval is contained within another. This is because of a result by Bogart and West [8], showing the equivalency of the corresponding interval graphs in the offline setting.

There has also been some work on multiple identical machines. For the case of equal-length, arbitrary-weight intervals, Fung et al. [19] give an algorithm that is 2-competitive when m, the number of machines, is even, and $(2 + \frac{2}{2m-1})$ when m is odd. Yu & Jacobson [38] consider C-benevolent (weight function is convex increasing) jobs and get an algorithm that is 2-competitive when m is even, and $(2 + \frac{2}{m})$-competitive when m is odd.

In the problem of call control, a graph is given, and requests that correspond to pairs of nodes of the graph arrive online. The goal is to accept as many requests as possible, with the final set consisting of disjoint paths. When the underlying graph is a line, this problem is closely related to ours. For call control on the line, Garay et al. [22] give optimal deterministic algorithms. In the unweighted case,

they achieve a $O(\log(n))$ competitive ratio, where n is the number of the vertices of the graph. In the case of proportional weights (weight is equal to the length of the path), they give an optimal algorithm that is $(\sqrt{5}+2) \approx 4.23$-competitive (its optimality was shown by Furst and Tomkins [36]). Adler and Azar [1] use randomization to overcome the $\log(n)$ lower bound, and give a 16-competitive algorithm. Emek et al. [15] study interval selection in the streaming model, and show how to modify their streaming algorithm to work online, achieving a competitive ratio of 6, improving upon the 16-competitive algorithm of Adler and Azar. It is noteworthy that the Adler and Azar algorithm uses memory proportional to the entire input sequence. In contrast, the Emek et al. algorithm only uses memory that is within a constant factor of a current OPT solution. It is still an open question if a randomized algorithm using only constant bounded memory can get a constant ratio in the unweighted case. We show that for a strict, but natural definition of memoryless randomized algorithms, a constant ratio cannot be obtained. The algorithms presented in this paper, along with the optimal algorithms by Garay et al. [22] and Woeginger [37], fall under our definition of memoryless. It is worth noting that similar notions of memoryless algorithms, and comparison between randomized memoryless and deterministic, have appeared in the k-server, caching, and facility location literature [12,14, 18,26,28,29,33]. We would note that barely random algorithms as described earlier (i.e. algorithms that initially generate some random bits, which are used in every online step), are not memoryless but usually satisfy bounded memory. The algorithms by Fung et al. [20] are an example of this. More generally, this use of initial random bits are the *classify and randomly select* algorithms (e.g. Lipton and Tomkins [31] and Awerbuch et al. [3]). It's important to note that such algorithms may require prior knowledge of bounds on lengths of intervals. In the full version of the paper [10] we discuss our meaning of memoryless and bounded memory online algorithms, and the relation to randomness, advice, and the Adler and Azar question.

The problem of admission control has also been studied under the model of minimizing rejections [2,7] instead of maximizing acceptances. An alternative input model for interval selection is that of arriving conflicts [23] instead of single intervals, with the algorithm being able to choose at most one item from each conflict. We also note that an instance of interval selection can be represented as an interval graph, with intervals corresponding to vertices, and edges denoting a conflict between two intervals. Generally, interval graphs reveal much less about the instance compared to receiving the actual intervals. In the interval graph representation, arriving vertices may have an adjacency list only in relation to already arrived vertices, or they may show adjacency to future vertices as well.

Our Results. For the unweighted adversarial case, we know that no deterministic algorithm is bounded (follows from [22]). Assuming there are at most k different lengths, we show how a simple greedy algorithm achieves a competitive ratio of $2k$. We also give a matching lower bound that holds for all deterministic algorithms, as well as "memoryless" randomized algorithms. We note that an instance with k different lengths can have a nesting depth of at most $k-1$. Alter-

natively, we can state our results in terms of d, the nesting depth (see Fig. 1), noting that $d \leq (k-1)$. This implies that our $2k$ bounds can be restated as $2(d+1)$. We also show how to extend the classify and randomly select paradigm used by Fung et al. [20] to obtain a randomized algorithm that is $2k$-competitive for the case of arbitrary weights and k different interval lengths. It's worth noting that the analysis by Canetti and Irani [11] implies an $\Omega(\sqrt{k})$ lower bound for randomized algorithms with arbitrary weights.

We show how the problem of call control on the line [22] relates to interval selection, and in particular how their $\log n$-competitive algorithm for the unweighted case and their $(2+\sqrt{5})$-competitive algorithm for proportional weights carries over to interval selection. In doing so, we explain why there is no contradiction between our optimal $2k$ bound, and their optimal $\log n$ bound. Lastly, we consider deterministic memoryless algorithms for the problem of any-order, unweighted, single-lengthed (i.e. unit) intervals with random order arrivals. We show that the only deterministic memoryless algorithm that can possibly perform better than the adversarial bound is one-directional, only replacing intervals if they overlap in that particular direction.

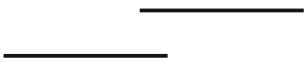
Organization of the Paper. Section 2 has some definitions to clarify the model. Section 3 has our upper and lower bounds in the adversarial case. Section 4 discusses arbitrary weights. Section 5 is about interval selection in the random order model. We end with some conclusions and open problems. The connection to call control, and the application of the proportional weights algorithm to our model can be found in the full version of the paper.

2 Preliminaries

Our 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. This allows two adjacent intervals $[s_1, f)$ and $[f, f_2)$ to not conflict, although our results would apply even if we considered closed intervals $[s_i, f_i]$ with $[s_1, f]$ and $[f, f_2]$ conflicting. There are two main ways two intervals can conflict, and they are shown in Fig. 1. In the case of containment, we say that the smaller intervals are *subsumed* by the larger one.

We use the notion of competitive ratio to measure the performance of our online algorithms. Given an algorithm A, let ALG denote the objective value of the solution achieved by the algorithm, and let OPT denote the optimal value achieved by an offline algorithm. The competitive ratio of A is defined as follows: $CR(A) = \frac{OPT}{ALG} \geq 1$. We should note that we can repeat disjoint copies of our nemesis sequences, and get the corresponding tight lower bounds. As a result, we can omit the standard additive term in our definition of competitive ratio. We will sometimes abuse notation and use ALG and OPT to denote the sets of intervals maintained by the algorithm at some given point, and the set of intervals of an optimal solution respectively. In the case of deterministic algorithms and random arrival of intervals, the performance of an algorithm is a random

variable, and the competitive ratios hold w.h.p. (definition of competitive ratio remains unchanged). The algorithm we present in the case of arbitrary weights is randomized, and its expected competitive ratio is defined as $CR(A) = \frac{OPT}{\mathbb{E}[ALG]}$.



(a)Partial Conflict.

(b)Containment with nesting depth 1.

Fig. 1. Types of conflicts.

We sometimes refer to a *chain* of intervals (Fig. 2). This is a set of intervals where each interval partially conflicts with exactly two other intervals, except for the two end intervals that partially conflict with only one.

Fig. 2. Interval chain.

3 Adversarial Order

3.1 Unweighted

In this section, we assume an adversary chooses the instance configuration, along with the arrival order of all intervals. Lemma 1 shows that revocable decisions are necessary even in the case of two different lengths. Algorithm 1 is the greedy algorithm that achieves the optimal competitive ratio of $2k$ in the unweighted case, and it works as follows: On the arrival of a new interval, take it if there's no conflict. If there's a conflict, take the new interval only if it is properly contained inside an existing interval.

Lemma 1. *The problem of any-order unweighted interval scheduling with two different lengths and irrevocable decisions is unbounded.*

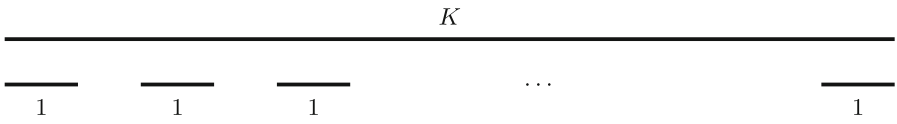


Fig. 3. Unweighted instance with two different lengths.

Algorithm 1.

```
On the arrival of I:
I_s ← Set of intervals currently in the solution conflicting with I
for I' ∈ I_s do
    if I ⊂ I' then
        Take I and discard I'
        return
    end if
end for
Discard I
```

Proof. Consider two possible interval lengths of 1 and K (Fig. 3). Let an interval of length K arrive first. W.l.o.g. the algorithm takes it (otherwise no smaller intervals arrive). Then K 1-length non-overlapping intervals arrive next, all of them overlapping with the first K-length interval. The algorithm cannot take any of the 1-length intervals, achieving a competitive ratio of $\frac{1}{K}$. This construction can be repeated multiple times.

Theorem 1. *Algorithm 1 achieves a competitive ratio of* $2k$ *for the problem of any-order unweighted interval scheduling with* k *different lengths.*

Proof. We define a mapping of intervals $f : OPT \longrightarrow ALG$, where every interval in ALG has at most $2k$ intervals in OPT mapped to it. Because intervals taken by the algorithm might be replaced during the execution, the mapping f might be redefined multiple times. What follows is the way optimal intervals $I \in OPT$ are charged, as soon as they arrive, to intervals $I' \in ALG$. There are four cases of interest:

Case 1: The newly arrived optimal interval is taken by the algorithm.
This can happen either because this interval did not conflict with any other intervals taken by the algorithm, or because it was entirely subsumed by a larger interval in ALG, in which case the algorithm would have replaced the large interval with the new small one. In this case, this optimal interval is mapped onto itself.

Case 2: The newly arrived optimal interval partially conflicts with one interval currently in ALG. In this case, this optimal interval is charged to the interval it conflicts with.

Case 3: The newly arrived optimal interval partially conflicts with two intervals currently in ALG. In this case, this optimal interval can be charged to either

of these two intervals arbitrarily. We may assume it is always charged to the interval it conflicts with on the right. Notice also, that a newly arrived interval, cannot partially conflict with more than two intervals in ALG.

Case 4: The newly arrived optimal interval subsumes an interval currently in ALG. W.l.o.g. we can assume this never happens. Any such optimal solution OPT can be turned into an optimal solution OPT', with the smaller interval in place of the larger one. We can restrict ourselves to only look at optimal solutions where no such transformation can take place. This case also encapsulates the case of an optimal interval perfectly coinciding with an interval taken by the algorithm.

An interval (I_l) taken by the algorithm can later be replaced, if a smaller one (I_s) comes along and is subsumed by it. When this happens, all intervals in OPT charged to I_l up to that point, will be transferred and charged to I_s. As a result, there are two ways an interval taken by the algorithm can be charged by intervals in OPT. The first way is when an interval $I \in OPT$ is directly charged to an interval $I' \in ALG$ when I arrives (Cases 1–4). This will be referred to as *direct charging*. The second way is when a new interval, I_n, arrives, and replaces an existing interval I_e, in which case all optimal intervals previously charged to I_e, will now be charged to I_n. This will be referred to as *transfer charging*.

Proposition 1. *An interval taken by the algorithm (even temporarily), can be charged by at most two optimal intervals through direct charging.*

To see why this proposition is true, we consider the three main cases of direct charging explained earlier. In *Case* 1, the optimal interval is taken by the algorithm and is charged to itself. Because no other optimal interval can conflict with it, we know this interval will never be directly charged again.

In *Cases* 2 and 3, direct charging happens because of the optimal interval partially conflicting with one or two intervals currently taken by the algorithm. Because an interval taken by the algorithm can partially conflict with at most two optimal intervals (one on each side), it can be charged twice at most.

Proposition 2. *An interval taken by the algorithm can be charged by at most* $2k - 2$ *optimal intervals through transfer charging.*

Consider a sequence of interval replacements by the algorithm, where all optimal intervals charged to an interval in the sequence are passed down to the next interval in the sequence. The last interval in that sequence will have accumulated all the optimal intervals charged to the previous intervals in that sequence. Because we consider k different lengths, such a sequence can have up to k intervals, participating in $k - 1$ transfer charging events. We also know that every interval in that sequence can be charged at most two optimal intervals through direct charging (Proposition 1) before being replaced. Consequently, assuming two additional charges are added to each interval in that sequence, the last (smallest) interval will be charged $2(k - 1)$ optimal intervals through transfer charging.

We have described a process during which every optimal interval is charged to an interval in ALG. By Propositions 1 & 2, we know that an interval in ALG, can be charged by $2k$ intervals in OPT at most. Therefore, our algorithm has a competitive ratio of $2k$ for the problem of unweighted interval selection with revocable decisions and k different possible interval lengths.

We now provide a matching lower bound, showing that no deterministic algorithm can do better.

Theorem 2. *No deterministic algorithm can achieve a competitive ratio better than $2k$ for the problem of unweighted interval selection with revocable decisions and k different lengths.*

Proof. At any point during the execution, the algorithm will have exactly one interval in its solution, while the size of the optimal solution will keep growing. We begin by describing how the main component of the instance is constructed, using intervals of the same length. First, the adversary must decide on an overlap amount v, which can be arbitrarily small. All partially conflicting intervals will overlap by exactly this amount. Consider now the instance of Fig. 4. Intervals I_1 and I_2 arrive first in that order. If I_1 is taken by the algorithm and is then replaced by I_2, then I_4 arrives. If I_1 was taken by the algorithm but was not replaced by I_2, then I_3 would arrive. Because this case is symmetrical, we only consider the former case of I_2 replacing I_1. What happens is that this chain keeps growing in the same direction, until the algorithm decides to stop replacing. When that happens, we look at the last three intervals of the chain. For example, when I_4 arrived, if the algorithm chose to not select I_4 and instead maintain I_2, we stop growing the chain and consider the intervals (I_1, I_2, I_4). If the algorithm never stops replacing, it will end up with I_5 in its solution. Notice that if the algorithms keeps replacing a growing chain, this will hurt the competitive ratio. In all cases, there exists an optimal solution of at least two intervals, with neither of them being the one taken by the algorithm. Note also that this construction requires at most four intervals of length L, occupying space at most $(4L - 3v)$ in total.

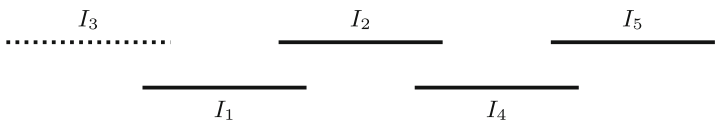


Fig. 4. Base adversarial construction

A small detail is that w.l.o.g. we can assume I_1 is always taken by the algorithm when it first arrives. Because this construction will take place a number of times during the execution, when the algorithm will already have an interval in its solution, it is useful to consider the case when I_1 is not taken by the algorithm. In this case, we start growing the chain regardless. If I_2 or I_4 are taken by

the algorithm, we treat it similarly to when I_1 was taken and the algorithm kept replacing. If the algorithm hasn't taken any interval even after I_4 has arrived, the chain stops growing and we consider the intervals (I_1, I_2, I_4).

Let I_{alg} be the interval taken by the algorithm (or I_2 if no intervals were taken). All remaining intervals to arrive will be subsumed by I_{alg}, and thus will not conflict with the two neighboring intervals taken by OPT. Assuming I_{alg} conflicts with one interval on the left and one on the right, that leaves space of length $(L - 2v)$ for all remaining intervals. Inside that space, the exact same construction described will take place, only when the algorithm takes a new interval, it implies I_{alg} is replaced. This can be thought of as going a level deeper, and using a sufficiently smaller interval length. More precisely, if L' is the new (smaller) length that will be used, it must hold that $L' \leq \frac{L+v}{4}$.

After each such construction is completed, the size of the optimal solution grows by at least 2. Because there are at most k different lengths, this can be repeated at most k times. Finally, because the algorithm only ever keeps a single interval in its solution, it will achieve a competitive ratio of $2k$.

We now extend Theorem 2 and show that the $2k$ lower bound also holds for a class of randomized algorithms we call *memoryless*. Intuitively, memoryless algorithms decide on taking or discarding the newly arrived interval, only by looking at the new interval, and all the intervals currently in the solution, using no information from previous online rounds. Although not randomized, it is worth noting that Algorithm 1, along with the optimal deterministic algorithms for call control [22], are memoryless.

Definition 1 (*Memoryless randomized algorithm*). *We call a randomized algorithm memoryless, if a newly arrived interval I_{new} is taken with probability $F(I_{new}, S)$, where $S = \{I_1, I_2, ...\}$ is the set of intervals currently in the solution, and each interval is a tuple of the form (s_i, f_i).*

Notice that Definition 1 only allows us to make use of random bits of this current step, and it does not allow access to random bits from previous rounds. In particular, this definition does not capture barely random algorithms (as mentioned in the introduction), or algorithms that fall under the *classify and randomly select* paradigm.

Theorem 3. *No memoryless randomized algorithm can achieve a competitive ratio better than $2k$ for the problem of unweighted interval selection with revocable decisions and k different lengths. More specifically, for all $p \in (0, 1]$, there exists an $\epsilon_p > 0$, such that the competitive ratio is greater than $2k - \epsilon_p$ with probability p.*

Proof. The proof is very similar to the proof of Theorem 2. The instance has the same structure as the one described in the proof of Theorem 2, with the difference that whenever a new interval is taken with probability $p > 0$, the adversary will have to add as many copies of that interval as necessary, so that it is taken w.h.p. Fig. 5 shows an example of multiple copies of a new interval, ensuring that a replacement happens w.h.p.

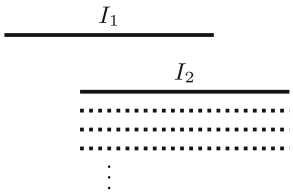


Fig. 5. Replacing I_1 w.h.p when $F(I_2, \{I_1\}) > 0$.

It is worth mentioning that similar to how we extend our lower bound to hold for memoryless randomized algorithms, one can extend the $\log(n)$ lower bound for call control [22] to also hold for memoryless randomized algorithms.

4 Arbitrary Weights

The case of intervals having an arbitrary weights has previously been considered for the case of single-length instances and ordered arrivals. Woeginger [37] gives an optimal deterministic algorithm that is 4-competitive. Fung et al. [20] give a barely random algorithm that is 2-competitive, and show that it is optimal amongst barely random algorithms that choose between two deterministic algorithms. Woeginger [37] shows that in the case of two different lengths, there does not exist a deterministic algorithm with finite competitive ratio. We show how to combine the barely random algorithm of Fung et al., with a classify and randomly select algorithm, to obtain a randomized algorithm for the any-order case, that achieves a competitive ratio of $2k$, when there are k different lengths.

First, one can observe that the 2-competitive single-length algorithm by Fung et al. [20] (Theorem 3.1), works even in the case of any-order arrivals. Our algorithm (denoted as ARB) works as follows: Choose one of k lengths, uniformly at random. Then execute the algorithm of Fung et al., looking only at intervals of the chosen length.

Theorem 4. *Algorithm ARB achieves a competitive ratio of* $2k$ *for the problem of any-order interval selection with* k *different lengths and arbitrary weights.*

Proof. Let $L_1, L_2, ..., L_k$ be all the different lengths of an instance. Associated with length L_i, is a sub-instance C_i, comprised only of the intervals of length L_i. Let OPT_i denote the weight of an optimal solution on sub-instance C_i. The expected performance of the algorithm can be bounded as follows:

$$\mathbb{E}[ALG] \geq \frac{1}{k}\frac{OPT_1}{2} + \frac{1}{k}\frac{OPT_2}{2} + ... + \frac{1}{k}\frac{OPT_k}{2} \geq \frac{OPT}{2k}$$

The first inequality holds because applying Fung et al. [20] on C_i gives a solution of weight at least $\frac{OPT_i}{2}$. The second inequality holds because for every length L_j, the total weight of the intervals of length L_j in the final solution, is at most OPT_j.

We note that the algorithm does not need to know the actual lengths beforehand, or even k. The algorithm can start working with the first length that appears. When a second length arrives, the algorithm discards its current solution and chooses the new length with probability $\frac{1}{2}$. More generally, when the ith length arrives, the algorithm starts over using the new length with probability $\frac{1}{i}$. One can see that the probability that any length is chosen is $\frac{1}{k}$. This procedure can be viewed as a form of reservoir sampling. Moreover, by replacing the 2-competitive arbitrary weights algorithm with a simple greedy algorithm, we get a randomized algorithm for the unweighted case that is $2k$-competitive and does not use revoking.

5 Random Order

In this section, we assume the adversary chooses the instance configuration, but the intervals arrive in a random order. We consider unweighted, single-lengthed instances, and deterministic memoryless algorithms with revocable acceptances. We consider various cases and show that the only type of algorithm that can beat the adversarial bound is a *one-directional* algorithm, namely an algorithm that only replaces intervals on the left side, or only on the right side, regardless of the amount of overlap. For any other algorithm, we show how the adversary can enforce a competitive ratio of 2, resulting in no benefit over adversarial arrivals for single-lengthed instances.

The argument works as follows. We first consider two simple algorithms, an algorithm that always replaces, and an algorithm that never replaces, and give a class of instances where these algorithms are no better than 2-competitive. We then show that for any algorithm that isn't one-directional, we can construct an instance on which the algorithm's behavior is the same as that of an always-replace, or never-replace algorithm on that class of instances. As a result, we get the following, arguably surprising, theorem. The proof of Theorem 5 is presented in the full version of the paper.

Theorem 5. *Every deterministic memoryless algorithm that isn't one-directional, can be forced to a strict competitive ratio of at least 2 for the problem of online unweighted single-lengthed interval selection under random order arrivals.*

We note that in ongoing work with additional authors, we have shown that for chain instances, the one-directional algorithm is substantially better than 2-competitive.

6 Conclusions and Open Problems

There are a number of possible directions for future work. A very natural direction is looking at specific weighted cases. Deterministically, Garay et al. [22] have settled the case of proportional weights with an optimal, constant-competitive

algorithm. It's interesting to see if a similar constant can be achieved for the more general weight functions studied by Woeginger [37], with or without randomness. We considered the case of arbitrary weights in Sect. 4.

It is fair to say that we have a limited understanding of randomized algorithms for interval selection. In the unweighted adversarial setting, we have shown that no memoryless randomized algorithm can be constant-competitive. With memory, the best known algorithm is 6-competitive, and we know of no better lower bound than $\frac{4}{3}$ (using a chain of 3 intervals). For some weighted cases (including proportional weights), Fung et al. show that with one random bit, their 2-competitive algorithm is optimal. However, these upper bounds don't necessarily hold in the any-order model. We also note that for arbitrary weights, there is a gap between our $2k$ upper bound, and the $\Omega(\sqrt{k})$ lower bound by Canetti and Irani [11]. We would like to extend the memoryless model to algorithms with constant memory beyond the current solution. In particular, we would want to allow access to a few initial random bits which would also capture algorithms that fall under the *classify and randomly select* paradigm.

A natural extension of revocable decisions is assigning a cost to the removal and replacement of previously accepted items, a model arguably more relevant in practice. This has been applied to problems such as online knapsack [24], and online advertising [4]. We find it interesting to consider interval selection with costs, and devise algorithms that aim to optimize the solution while limiting the total cost of revoking.

Finally, to the best of our knowledge, we have initiated the study of this model under random order arrivals, where there are many open questions for future work. We have only looked at single-lengthed instances, a special case that, in the adversarial setting, doesn't even require revoking. Looking at multiple lengths under random arrivals is a natural next step. Lastly, we have shown that one-directional algorithms for single-lengthed instances, are the only type of deterministic memoryless algorithms that can achieve better than 2−competitiveness. We don't have any provable upper bounds on the performance of a one-directional algorithm, but we have conducted experiments that suggest it may achieve much better than 2-competitiveness. This is an interesting contrast with the adversarial model, where a one-directional algorithm would perform arbitrarily bad.

Acknowledgements. We would like to thank Denis Pankratov, Adi Rosén and Omer Lev for many helpful comments.

References

1. Adler, R., Azar, Y.: Beating the logarithmic lower bound: randomized preemptive disjoint paths and call control algorithms. J. Scheduling **6**(2), 113–129 (2003)
2. Alon, N., Azar, Y., Gutner, S.: Admission control to minimize rejections and online set cover with repetitions. In: Proceedings of the Seventeenth Annual ACM Symposium on Parallelism in Algorithms and Architectures, pp. 238–244 (2005)

3. Awerbuch, B., Bartal, Y., Fiat, A., Rosén, A.: Competitive non-preemptive call control. In: SODA, vol. 94, pp. 312–320. Citeseer (1994)
4. Babaioff, M., Hartline, J.D., Kleinberg, R.D.: Selling ad campaigns: online algorithms with cancellations. In: Proceedings of the 10th ACM Conference on Electronic Commerce, pp. 61–70 (2009)
5. Bachmann, U.T., Halldórsson, M.M., Shachnai, H.: Online selection of intervals and t-intervals. Inf. Comput. **233**, 1–11 (2013)
6. Baptiste, P.: Scheduling equal-length jobs on identical parallel machines. Discret. Appl. Math. **103**(1–3), 21–32 (2000)
7. Blum, A., Kalai, A., Kleinberg, J.: Admission control to minimize rejections. In: Dehne, F., Sack, J.-R., Tamassia, R. (eds.) WADS 2001. LNCS, vol. 2125, pp. 155–164. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44634-6_15
8. Bogart, K.P., West, D.B.: A short proof that "proper= unit." Discret. Math. **201**(1–3), 21–23 (1999)
9. Borodin, A., Ivan, I., Ye, Y., Zimny, B.: On sum coloring and sum multi-coloring for restricted families of graphs. Theor. Comput. Sci. **418**, 1–13 (2012)
10. Borodin, A., Karavasilis, C.: Any-order online interval selection. arXiv preprint arXiv:2303.06127 (2023)
11. Canetti, R., Irani, S.: Bounding the power of preemption in randomized scheduling. In: Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, pp. 606–615 (1995)
12. Christou, D., Fotakis, D., Koumoutsos, G.: Memoryless algorithms for the generalized k-server problem on uniform metrics. In: Kaklamanis, C., Levin, A. (eds.) WAOA 2020. LNCS, vol. 12806, pp. 143–158. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-80879-2_10
13. Chrobak, M., Dürr, C., Jawor, W., Kowalik, Ł, Kurowski, M.: A note on scheduling equal-length jobs to maximize throughput. J. Scheduling **9**(1), 71–73 (2006)
14. Coester, C., Koutsoupias, E.: The online k-taxi problem. In: Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (2019)
15. Emek, Y., Halldórsson, M.M., Rosén, A.: Space-constrained interval selection. ACM Trans. Algorithms (TALG) **12**(4), 1–32 (2016)
16. Epstein, L., Levin, A.: Improved randomized results for that interval selection problem. In: Halperin, D., Mehlhorn, K. (eds.) ESA 2008. LNCS, vol. 5193, pp. 381–392. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87744-8_32
17. Faigle, U., Nawijn, W.M.: Note on scheduling intervals on-line. Discret. Appl. Math. **58**(1), 13–17 (1995)
18. Fotakis, D.: On the competitive ratio for online facility location. Algorithmica **50**(1), 1–57 (2008)
19. Fung, S.P., Poon, C.K., Yung, D.K.: On-line scheduling of equal-length intervals on parallel machines. Inf. Process. Lett. **112**(10), 376–379 (2012)
20. Fung, S.P., Poon, C.K., Zheng, F.: Improved randomized online scheduling of intervals and jobs. Theory Comput. Syst. **55**(1), 202–228 (2014)
21. Garay, J.A., Gopal, I.S.: Call preemption in communication networks. In: Proceedings IEEE INFOCOM 1992: Conference on Computer Communications (1992)
22. Garay, J.A., Gopal, I.S., Kutten, S., Mansour, Y., Yung, M.: Efficient on-line call control algorithms. J. Algorithms **23**(1), 180–194 (1997)
23. Halldórsson, M.M., Patt-Shamir, B., Rawitz, D.: Online scheduling with interval conflicts. Theory Comput. Syst. **53**(2), 300–317 (2013)

24. Han, X., Kawase, Y., Makino, K.: Online Knapsack problem with removal cost. In: Gudmundsson, J., Mestre, J., Viglas, T. (eds.) COCOON 2012. LNCS, vol. 7434, pp. 61–73. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32241-9_6
25. Hyatt-Denesik, D., Rahgoshay, M., Salavatipour, M.R.: Approximations for throughput maximization. arXiv preprint arXiv:2001.10037 (2020)
26. Kleinberg, J.M.: A lower bound for two-server balancing algorithms. Inf. Process. Lett. **52**(1), 39–43 (1994)
27. Kolen, A.W., Lenstra, J.K., Papadimitriou, C.H., Spieksma, F.C.: Interval scheduling: a survey. Nav. Res. Logist. (NRL) **54**(5), 530–543 (2007)
28. Koutsoupias, E.: The k-server problem. CS Rev. **3**(2), 105–118 (2009)
29. Koutsoupias, E., Taylor, D.S.: The CNN problem and other k-server variants. Theor. Comput. Sci. **324**(2–3), 347–359 (2004)
30. Kovalyov, M.Y., Ng, C.T., Cheng, T.E.: Fixed interval scheduling: models, applications, computational complexity and algorithms. Eur. J. OR (2007)
31. Lipton, R.J., Tomkins, A.: Online interval scheduling. In: SODA, vol. 94 (1994)
32. Miyazawa, H., Erlebach, T.: An improved randomized on-line algorithm for a weighted interval selection problem. J. Sched. **7**(4), 293–311 (2004)
33. Raghavan, P., Snir, M.: Memory versus randomization in on-line algorithms. In: Ausiello, G., Dezani-Ciancaglini, M., Della Rocca, S.R. (eds.) ICALP 1989. LNCS, vol. 372, pp. 687–703. Springer, Heidelberg (1989). https://doi.org/10.1007/BFb0035792
34. Seiden, S.S.: Randomized online interval scheduling. Oper. Res. Lett. **22**(4–5), 171–177 (1998)
35. Sgall, J.: On-line scheduling. Online Algorithms, pp. 196–231 (1998)
36. Tomkins, A.: Lower bounds for two call control problems. Inf. Process. Lett. **56**(3), 173–178 (1995)
37. Woeginger, G.J.: On-line scheduling of jobs with fixed start and end times. Theor. Comput. Sci. **130**(1), 5–16 (1994)
38. Yu, G., Jacobson, S.H.: Online c-benevolent job scheduling on multiple machines. Optim. Lett. **12**(2), 251–263 (2018)