Online Interval Scheduling

Hailiang Zhao
Zhejiang University

March 5, 2024

A lecture based on the paper —— R. Lipton et al., Online interval scheduling. In: SODA '94.

Outline

Definition

An interval I is a pair of positive real numbers

$$\langle \mathsf{start}(I), \mathsf{len}(I) \rangle.$$

We also refer to $\mathsf{len}(I)$ as $|I|$.

For a positive real number r we say $r \in I$ if

$$\mathsf{start}(I) \leq r \leq \mathsf{start}(I) + |I|. \tag{1}$$

A problem instance (PI) is a finite set S of intervals to be scheduled. No two intervals in S share a start point.

We say $\sigma \subset S$ is a feasible schedule (FS) if no two intervals of σ overlap.

Definition

For a FS σ, we define the weight of σ as

$$|\sigma| := \sum_{I \in \sigma} |I|. \tag{2}$$

Let σ^* be some maximum weight FS of S. Let $\mathbf{F}(S)$ be the collection of all FSs of S.

A randomized scheduling algorithm (RSA) A takes a PI S and returns A(S), a FS of S. We treat A(S) as a random variable over $\mathbf{F}(S)$.

The weight of A on S is the expected weight of the FS returned by A:

$$W(\mathsf{A}, S) = \mathbb{E}[W(\mathsf{A}(S))]. \tag{3}$$

Definition

A RSA A is *online* if for any interval $I \in S$, $\Pr[I \in \mathsf{A}(S)]$ depends only on intervals with start times before start(I).

We say A is c-competitive if

$$\forall S : c \cdot W(\mathsf{A}, S) \geq |\sigma^*|. \tag{4}$$

Explanation:

$$\frac{1}{W(\mathsf{A}, S)} / \frac{1}{|\sigma^*|} \leq c \Leftrightarrow \frac{|\sigma^*|}{W(\mathsf{A}, S)} \leq c \Leftrightarrow c \cdot W(\mathsf{A}, S) \geq |\sigma^*|.$$

A is strongly c-competitive if A is c-competitive, and there is no b-competitive algorithm with $b < c$ (the smaller, the better).

Definition

The weight of A on a particular interval $I \in S$ is defined as

$$W_S(\mathsf{A}, I) := |I| \cdot \Pr[I \in \mathsf{A}(S)]. \tag{5}$$

Lemma 1

$W(A, S) = \sum_{I \in S} W_S(A, I)$.

Proof.

The proof is based on the linearity of expectation:
$W(\mathsf{A}, S) = \mathbb{E}[W(\mathsf{A}(S))] = \sum_{\sigma \in \mathbf{F}(S)} |\sigma| \cdot \Pr[\mathsf{A}(S) = \sigma] =$
$\sum_{I \in S} |I| \cdot \Pr[I \in \mathsf{A}(S)] = \sum_{I \in S} W_S(\mathsf{A}, I)$. □

The weight of A on a subset of S is defined by: If $T \subset S$, then

$$W_S(\mathsf{A}, T) := \sum_{I \in T} W_S(\mathsf{A}, I). \tag{6}$$

Outline

Definition

Let's consider PIs with only two types of intervals, small intervals of length 1 and large intervals of length $k \gg 1$. We name this kind of PIs as 2-Length PIs.

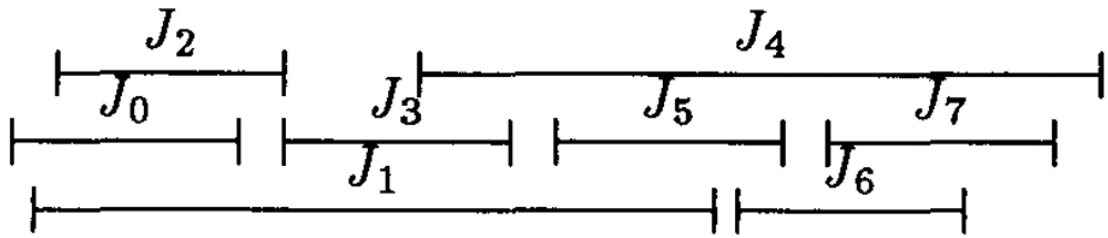


Figure 1: A 2-length PI. The intervals are indexed by their start times.

The Virtual Algorithm

It's easy to see that, to maximize the weight greedily, any time a k-interval arrives, we should take it whenever possible.

However, upon the arrival of a 1-interval, it's rational that sometimes we take it and sometimes we just drop it. We drop it because we expect that the saved resource *can be taken by a potential k-interval*. Thus, we may propose an algorithm like this:

The Virtual Algorithm (VA)

If the resource is in use, do nothing. Otherwise:

1. For a 1-interval, flip a fair coin. If head, take the interval; if tail, virtually take the interval, but *do no work* —— Take no 1-interval for the next 1 unit of time.
2. For a k-interval, take whenever possible.

Bucket

To establish the connection between VA and σ^*, we introduce *bucket* —— A bucket is a set of intervals (or parts of intervals), and for each $J \in \sigma^*$, we associate a bucket with it, referred to as bucket(J). Each interval in σ^* is assigned to its own bucket.

For any subinterval I', let the interval I containing I' be called Parent(I'). Then, for any bucket, we define

$$W_S(\mathsf{A}, \text{bucket}) := \sum_{I' \in \text{bucket}} |I'| \cdot \Pr[\text{Parent}(I') \in \mathsf{A}(S)]. \quad (7)$$

In other words, for each bucket, we only count the length of subintervals that are assigned to it. We say an interval I *blocks* another interval J if I overlaps start(J).

Assigning Intervals to Buckets

Lemma 2

For intervals of S can be assigned to buckets such that:

1. *no part of any interval is placed in more than one bucket, and*
2. $\forall J \in \sigma^*$, $W_S(A, \mathit{bucket}(J)) \geq \frac{|J|}{2}$,

then A is 2-competitive.

The proof is simple ——

$$
\begin{aligned}
W(\mathsf{A}, S) &= \sum_{I \in S} W_S(\mathsf{A}, I) && \text{by Lemma 1} \\
&\geq \sum_{J \in \sigma^*} W_S(\mathsf{A}, \mathsf{bucket}(J)) && \text{by (7) and condition 1} \\
&\geq \sum_{J \in \sigma^*} \frac{|J|}{2} = \frac{|\sigma^*|}{2}. && \text{by condition 2} \qquad (8)
\end{aligned}
$$

Analysis Framework

If (i) we can construct a way to assign intervals to buckets such that the first condition holds, and (ii) we prove that under this assignment the second condition holds for VA, we then prove VA is 2-competitive.

First, let's consider how to assign intervals to buckets.

Assigning Intervals in σ^*

Each interval in σ^* is assigned to its own bucket.

Then, we address individually the assignments of 1-intervals $I \notin \sigma^*$ and k-intervals $B \notin \sigma^*$ to buckets.

Assigning 1-Intervals that are not in σ^*

Assigning 1-Intervals

Any 1-interval can block at most one interval of σ^*. Thus, for each 1-interval of S, place I in the bucket of the interval of σ^* that it blocks, if any.

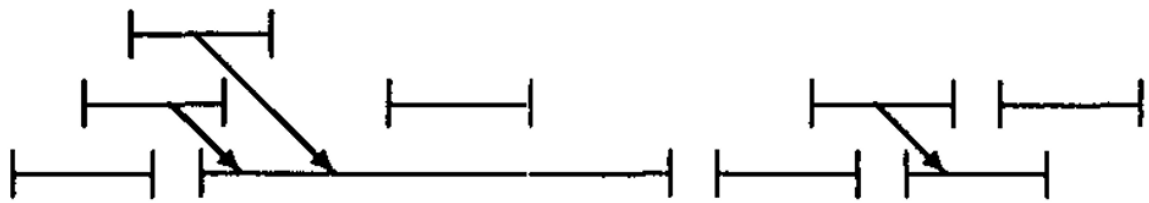


Figure 2: σ^* is shown along the bottom row. The other elements of S are in the upper rows. All elements of σ^* are in their own buckets. Each other element of S has an arrow showing which bucket it is assigned to, if any.

Assigning k-Intervals that are not in σ^*

We say $J \in \sigma^*$ is the terminal interval of $I \in S$ if J contains the end point of I. An interval will have at most one terminal interval because otherwise two intervals in σ^* would overlap. Some intervals will not have a terminal interval.

We say a k-interval B covers an interval I if start(I) and start$(I) + |I|$ are both contained in B.

Assigning k-Intervals that are not in σ^*

Assigning k-Intervals

Say T is the terminal interval of B. We will assign *at least half the length of* B to (the bucket of) T. If T does not exist, the parts of B that would have been assigned to T are not assigned to any bucket.

If B covers an interval J of σ^*, we do the following: since J must have length 1, there is a corresponding 1 unit of length of B which covers J. Assign the first half of this length to J, and the second half of it to T. Assign all unassigned subintervals of B to T.

Assigning k-Intervals that are not in σ^*

Below is an example of assigning a k-interval.

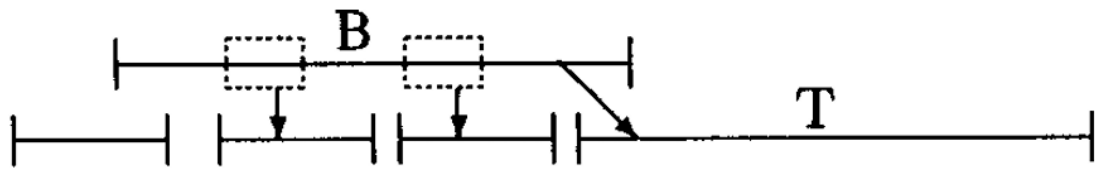


Figure 3: The terminal interval of B is labelled with a T. All subintervals of B which are not assigned to T are marked with dashed boxes, and have arrows showing the element of σ^* to which they are assigned.

Note that there are at most k 1-intervals that can be covered by B. Even if all the k 1-intervals are in σ^* (which is actually impossible if intervals are closed), at most $\frac{k}{2}$-length of subintervals are assigned to them. Thus, there are at least $\frac{k}{2}$-length of subintervals can be assigned to T, if any.

The First Condition of Lemma 2 Holds

It's easy to see that the first condition of Lemma 2 is obeyed:

- 1-intervals can never block more than one interval of σ^*.
- Any part of a k-interval that is assigned to a bucket will be assigned either to the bucket of its terminal interval or to that of the overlapping interval of σ^*.

What About the Second Condition?

We restate and prove the second condition of Lemma 2.

Lemma 3
For all intervals $J \in \sigma^*$,

$$W_S(\mathit{VA}, \mathit{bucket}(J)) \geq \frac{|J|}{2}. \tag{9}$$

We can prove this by separately prove the contribution from the bucket of 1-intervals and the bucket of k-intervals, i.e., to show that for any 1-interval $I \in \sigma^*$, $W_S(\mathsf{VA}, \mathsf{bucket}(I)) \geq \frac{1}{2}$, and for any k-interval $B \in \sigma^*$, $W_S(\mathsf{VA}, \mathsf{bucket}(B)) \geq \frac{k}{2}$.

Proof for 1-Intervals

We first consider the buckets associated with 1-intervals of σ^*.

Let $I \in \sigma^*$ be any 1-interval of σ^*. We then partition $\mathbf{F}(S)$ into four disjoint groups of FSs based on *the treatment of I in the schedule*.

The First Group

The first group comprises those FSs with the property that the resource is available (neither taken or virtually taken) immediately prior to the start of some 1-interval I' that blocks I —— I' is able to be scheduled by VA with probability $\frac{1}{2}$.

Thus, we have

$$\begin{aligned} W_S(\mathsf{VA}, \mathsf{bucket}(I)) &= \sum_{I' \in \mathsf{bucket}(I)} |I'| \cdot \Pr[\mathsf{Parent}(I') \in \mathsf{VA}(S)] \\ &= |I'| \cdot \Pr[I' \in \mathsf{VA}(S)] \\ &= 1 \cdot \frac{1}{2}. \end{aligned} \tag{10}$$

A FS from this group will be scheduled with probability p_1 for some unknown value of p_1.

The Second Group

The second group comprises those FSs in which VA schedules a k-interval B (with probability 1) that covers I. By our assignment rule, a subinterval of B with a length of $\frac{1}{2}$ is assigned to bucket(I).

Thus, the weight from bucket(I) for any FS in this group is at least $\frac{1}{2}$.

A FS from this group will be scheduled with probability p_2 for some unknown value of p_2.

The Third Group

The third group comprises those FSs in which a k-interval is scheduled for which I is the terminal interval, i.e., the k-interval ends in I.

Thus, the weight from bucket(I) for any FS in this group is at least $\frac{k}{2}$, since at least half the length of the k-interval will be assigned to bucket(I).

A FS from this group will be scheduled with probability p_3 for some unknown value of p_3.

The Fourth Group

The fourth group comprises those FSs in which the resource is neither taken nor virtually taken at the start of I.

Since I is in its own bucket, the weight from bucket(I) for any FS in this group is at least $\frac{1}{2}$.

A FS from this group will be scheduled with probability $p_4 = 1 - p_1 - p_2 - p_3$.

Proof for 1-Intervals

Thus, the gain of VA on bucket(I) is at least

$$p_1 \cdot \frac{1}{2} + p_2 \cdot \frac{1}{2} + p_3 \cdot \frac{k}{2} + p_4 \cdot \frac{1}{2} \geq \frac{1}{2}, \tag{11}$$

which induces that $W_S(A, I) \geq \frac{|I|}{2}$.

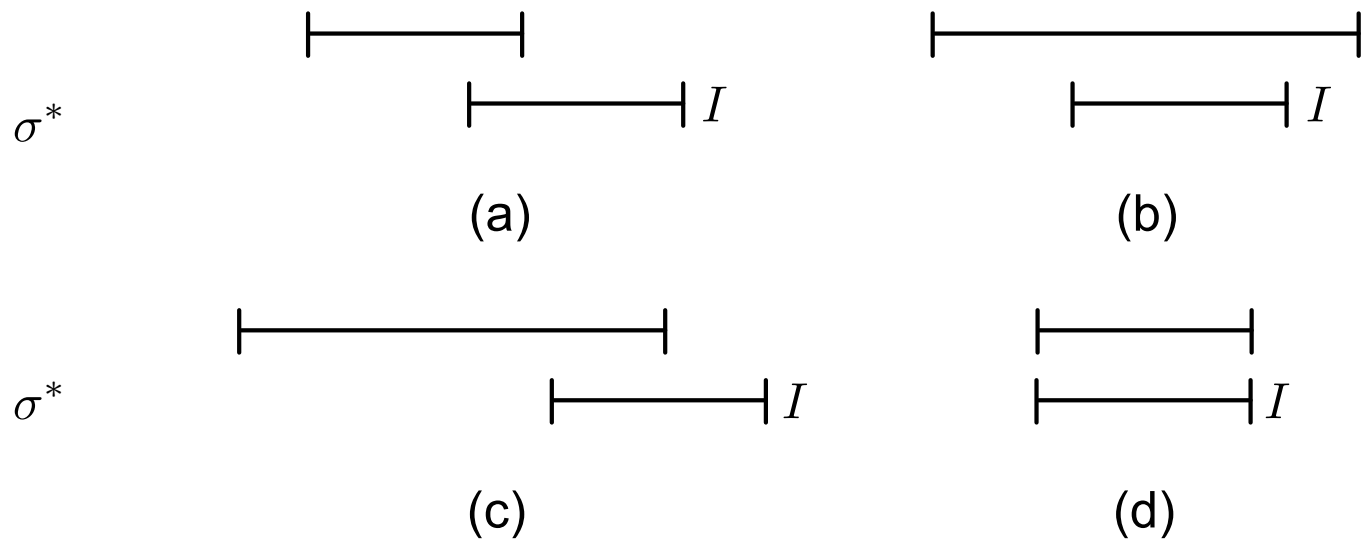


Figure 4: The four cases w.r.t. a 1-interval in σ^*.

Proof for k-Intervals

Assume $B \in \sigma^*$ is any k-interval. There are three different groups of $\mathbf{F}(S)$ w.r.t. the treatment of B.

The First Group

The first group comprises those FSs in which the resource schedules a k-interval B' overlapping B.

Since B is the terminal interval of B', at least half the length of B' is assigned to B. Thus, the weight of VA on bucket(B) is at least $\frac{k}{2}$.

This case occurs with probability p_1.

The Second Group

The second group comprises those FSs with the property that the resource is free immediately preceding the start of some 1-interval I that blocks B.

With probability $\frac{1}{2}$, the 1-interval will be scheduled, and with probability $\frac{1}{2}$, the 1-interval will be virtually scheduled, allowing us to schedule a k-interval (either B or another k-interval in bucket(B)).

The weight of bucket(B) in this case is thus $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \frac{k}{2} \geq \frac{k}{2}$.

This case occurs with probability p_2.

The Third Group

The third group comprises those FSs which leave the resource free (neither taken nor virtually taken) immediately prior to the start of B, or if B is the terminal interval of some k-interval $B' \in S$, immediately prior to the start of B'.

Since all of B and at least half the length of B' must be assigned to the bucket of B, the weight of VA on bucket(B) must be at least $\frac{k}{2}$.

This occurs with probability $p_3 = 1 - p_1 - p_2$.

The difference between Case 1 and Case 3 —— In Case 1, the resource is already taken while in Case 3 the resource is neither taken nor virtually taken, in which probability involves.

Proof for k-Intervals

Thus, the gain of VA on bucket(B) is at least

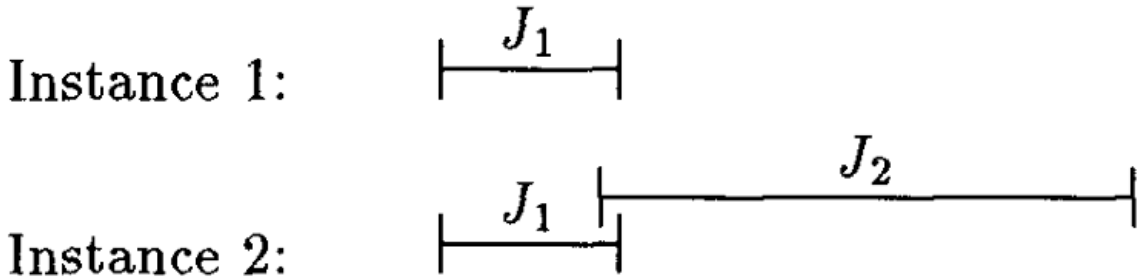
$$p_1 \cdot \frac{k}{2} + p_2 \cdot \frac{k}{2} + p_3 \cdot \frac{k}{2} = \frac{k}{2}, \tag{12}$$

which induces that $W_S(A, B) \geq \frac{|B|}{2}$.

With the proof for 1-interval and k-intervals, we completes the proof of Lemma 2.

Lower Bound of the Competitive Ratio

Consider the following 2-length PIs:



Let $p_1 = \Pr[\text{A takes } J_1]$ and $p_2 = \Pr[\text{A takes } J_2]$. We denote by S_1 and S_2 the set of intervals in Instances 1 and 2, respectively.

If $p_1 < \frac{1}{2}$, $W(A, S_1) = p_1 \cdot 1 < \frac{1}{2} = 2 \cdot \sigma^*(S_1)$, i.e., A is not even 2-competitive. So we assume $p_1 \geq \frac{1}{2}$. Thus $p_2 < \frac{1}{2}$ since J_1 and J_2 cannot be scheduled simultaneously. Then $2 \cdot W(A, S_2) = 2(p_1 + p_2 \cdot k) \approx 2p_2k < k = \sigma^*(S_2)$, which means A can be no better than 2-competitive on Instance 2.

Outline

The Online Marriage Problem

There is a game between a player and a host. The host picks a number n in advance, which is kept secret from the player. Each time the host present a number from the sequence $2, 2^2, ..., 2^n$ to the player. The game is played until

- either the player says "yes" (in which the player gets the money) or
- the player refuses n requests (in which case the player gets nothing).

Formally, we are given a finite geometric sequence $[b, b^2, ..., b^n]$. We seek a randomized algorithm A which, when offered $\$b^k$, flips a coin and says "yes" or "no" with probability p_k.

The Online Marriage Problem

We define a sequence c_i similar to p_i as follows:

$$c_i = p_i \prod_{j=1}^{i-1}(1 - p_j). \tag{13}$$

p_i is the probability that b^i is taken, given that all previous offers are not taken, while c_i is simply the probability that b^i is taken.

We define the gain G of algorithm A as a function of n:

$$G(A, n) = \sum_{i=1}^{n} c_i b^i. \tag{14}$$

The $(1+\epsilon)$-Algorithm

Recall the the *zeta function* is defined as follows:

$$\zeta(d) = \sum_{n=1}^{\infty} \frac{1}{d^n}. \tag{15}$$

For $d > 1$, $\zeta(d)$ converges.

The $(1+\epsilon)$-Algorithm

The algorithm flips coins such that

$$c_i = \frac{1}{i^{1+\epsilon}\zeta(1+\epsilon)}. \tag{16}$$

Obviously, $\sum_{i=1}^{\infty} c_i = \frac{1}{\zeta(1+\epsilon)} \sum_{i=1}^{\infty} \frac{1}{i^{1+\epsilon}} = \frac{1}{\zeta(1+\epsilon)} \cdot \zeta(1+\epsilon) = 1$.
Thus, $\{c_i\}$ represents a probability distribution over the b^i's.

The $(1+\epsilon)$-Algorithm

Note that p_i's can be back-solved from c_i and $p_1, ..., p_{i-1}$:

$$p_i = \frac{c_i}{\prod_{j=1}^{i-1}(1-p_j)}. \tag{17}$$

$[p_i]$ are used to present an algorithm that actually flips coins.

The $(1+\epsilon)$-Algorithm

Lemma 4
The $(1+\epsilon)$-algorithm is $O(n^{1+\epsilon})$-competitive.

Proof.
If the sequence has length n, OPT $= b^n$. Meanwhile, the algorithm will take b^n offer with probability c_n, so will have gain of at least $c_n b^n = O(\frac{b^n}{n^{1+\epsilon}})$. □

Outline

The Marriage Algorithm

We can extend the approach of VA to intervals of arbitrary lengths —— If we reject an interval I, schedule no other interval that begins during I unless it's *twice* as long as I.

We take the intervals whose length increase exponentially, as in the Marriage problem.

The Marriage Algorithm

Define the active interval I_{active} to be the interval which was most recently either taken or virtually taken. The depth of I_{active} is some positive integer calculated during the algorithm.

Initially, I_{active} is empty. Upon the arrival of a new interval I, decide as follows.

1. If I starts outside I_{active}, set depth to 1. Set I_{active} as I and take I with probability p_{depth}, o.w. virtually take I.
2. If I starts within I_{active} and I_{active} is taken, do nothing.
3. If I starts within I_{active}, I_{active} is virtually taken, and $|I| < 2|I_{\text{active}}|$, do nothing.
4. If I starts within I_{active}, I_{active} is virtually taken, and $|I| \geq 2|I_{\text{active}}|$, then set I_{active} as I, increment depth, and take I with probability p_{depth}, o.w. virtually take I.

The used sequence of probabilities $[p_i]$ is defined in (17).

Analyzing The Marriage Algorithm

Theorem 5

MA is $O((\log \Delta)^{1+\epsilon})$-competitive, where Δ is the ratio of longest to shortest intervals in S.

Proving the theorem is our main target.

Simplifying the Instances

To simplify the proof, we assume that all "free space" has been removed from S, i.e., start times have been modified to remove any time periods with no pending requests, without changing lengths, relative start orderings, or overlaps.

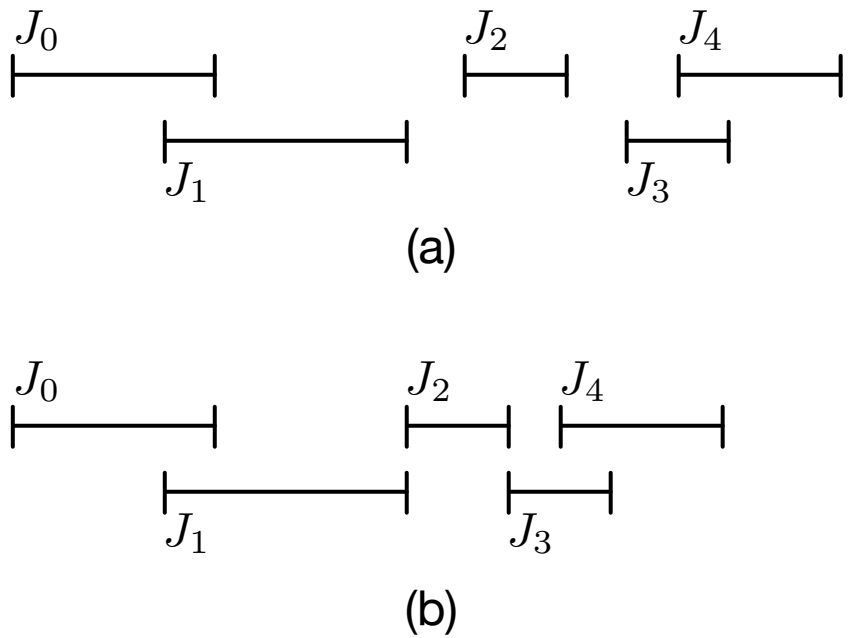


Figure 5: How we remove the "free space".

The operation clearly does not change the performance of algorithms in any way.

Constructing S_i

For a set of intervals $S_i \subset S$, define $|S_i|$ to be the distance between the start point of the first interval and the end point of the last-ending interval in S_i.

From the beginning of S construct the longest possible set of overlapping intervals $I_1, ..., I_m$ with the property that I_1 is the first interval in S and

$$\text{start}(I_j) < \text{start}(I_{j+1}) \tag{18}$$

and

$$|I_{j+1}| \geq 2|I_j|. \tag{19}$$

Call this set S_1. Note that S_1 is the longest possible set constructed in this way.

Constructing S_i

To create S_2 after the end of S_1, we eliminate "interference" from S_1.

Recall that an interval I is given the opportunity to be scheduled with some probability unless

1. either the resource is already taken (I is blocked), or
2. the resource is virtually taken by an interval more than half as long (I is virtually blocked).

Constructing S_i

Some intervals will have start points overlapping S_1, but will extend beyond the end of S_1. We look for the first interval J starting outside S_1 such that if I overlaps start(J) then either I begins outside S_1 or $2|I| < |J|$.

That is, we look for the first interval starting beyond the end of S_1 that could never be virtually blocked by any of these overlapping intervals.

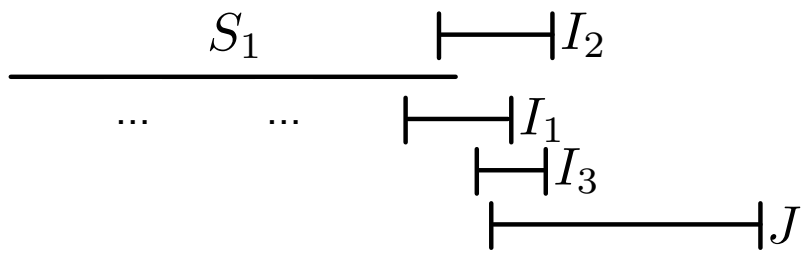


Figure 6: J is the chosen interval to be the beginning of S_2.

Once we locate J, we use it as the first interval of S_2, using the same construction as for S_1. We continue the process until the end of S.

Analysis Framework

We break S into regions corresponding to the S_i's and analyze the performance of MA on each region. An interval is placed in the region of a particular S_i if its start point occurs after the end of all intervals in S_{i-1} but before the end of the last interval of S_i.

Lemma 6

The weight of MA on the region of S_i is at least $\frac{c_{\log}\Delta}{4}|S_i|$.

This lemma indicates that MA performs well on each S_i. In the following, we present its proof.

Backtracking

Fix i. Let I be the last interval in S_i. Let σ be a FS generated by MA.

Knowing the entire problem instance S allows us to determine the state of the algorithm at various points of σ. For instance, consider the start point of I. We can identify this point in σ. Imagine moving back in time along σ until we reach the endpoint of some interval in σ. Call this interval L.

Since we know that the resource was free immediately after L, we know the state of MA, so we can simulate the algorithm from the endpoint of L onwards.

Backtracking

The interval $N \in S$ with the next start point would be presented to MA. A p_1-biased coin would be flipped.[1]

For the particular run of MA that generated σ, we note that if $N \in \sigma$ then the coin must have come up heads; o.w. the coin must have come up tails.

Whichever is the case, we can continue our simulation, determining the outcome of coin flips by checking to see if the interval in question appears in σ.

[1] Here depth $= 1$ since N starts outside L.

Backtracking

Eventually, we will be presented with the start point of I. At this point, we end the simulation. We can look back through the simulation to determine when the resource was *last* free (neither taken or virtually taken).

Immeidately after the last free point, MA would have flipped a p_1-biased coin for some interval J, and the resource would remain either taken or virtually taken until the start of I.

We focus our attention on interval J, which is referred to as *the leading interval of* I. For some FSs, I will be its own leading interval.

Leading Interval

Intuitively, the leading interval of I is the first interval for which a p_1-biased coin was flipped, as we proceed backwards from the start point of I.

We now break $\mathbf{F}(S)$ into groups that share the same leading interval of I. We call these groups *partitions* —— all schedules in a particular partition have the same leading interval.

Leading Interval

Intuitively, the leading interval of I is the first interval for which a p_1-biased coin was flipped, as we proceed backwards from the start point of I.

We now break $\mathbf{F}(S)$ into groups that share the same leading interval of I. We call these groups *partitions* —— all schedules in a particular partition have the same leading interval.

Consider a fixed partition P with leading interval J. For particular schedule $\sigma \in P$, MA might have flipped heads for J, in which case $J \in \sigma$, or might have flipped tails in which case $J \notin \sigma^*$. If MA flipped tails, J would have been virtually taken, and another interval might have had the opportunity to be scheduled.

From J to I

For any possible outcome of the coin flips, there will be some schedule in P corresponding to that outcome. Consider the schedule which results if, when faced with J, MA had actually flipped tails. And from that point on, MA flipped tails for every single coin flip it made.

Then one of two cases would occur:

1. MA would flip a coin for interval I.
2. MA, when presented with interval I, would not flip a coin because it had virtually scheduled some I' with length at least $\frac{|I'|}{2}$.

From J to I

If partition P lies in case 1, we would expect a random schedule from P to contain I with probability at least $c_{\log \Delta}$.[2]

If partition P lies in case 2, we would expect a random schedule from P to contain I' with probability also at least $c_{\log \Delta}$.

In both cases, $c_{\log \Delta}$ is a lower bound on the probability that MA, when starting at the leading interval of P, flips all tails until faced with I (or I'), and then flips heads.

[2] Note that c_i is the probability that b^i is taken, which embeds the probabilities that $b, \ldots, b^{i-1}$ are not taken. Since ratio of length $|I|/|J|$ is at most Δ, we need at least $\log_2 \Delta$ steps to reach I from J (in which the latter interval is just twice as long as the former one).

Scheduled Length Comparison

Each partition represents all possible outcomes of a particular online Marriage game —— the "offers" made to the player are the intervals for which coins would be flipped, beginning with the leading interval of the partition and continuing through either I or I'.

Since $2|I'| > |I|$, we expect to schedule at least $c_{\log \Delta} \frac{|I|}{2}$ from every partition. Intervals I and I' are both lie within the region associated with S_i ——

1. this is true for I since $I \in S_i$ by definition, and
2. for I' because I' virtually blocks an element of S_i, i.e., I, so could not also overlap an interval in S_{i-1}.[3]

Thus, the expected contribution from the region of S_i is at least $c_{\log \Delta} \frac{|I|}{2}$.

[3] O.w. I cannot be in the next region S_i.

Scheduled Length Comparison

Since each interval of S_i must be at least twice as long as the previous one, and each one overlaps the previous one, and I is the last interval of S_i, it must be the case that

$$|I| \geq \frac{|S_i|}{2}. \tag{20}$$

So, in both cases, we expect to acquire at least $\frac{c_{\log \Delta}}{4}|S_i|$ from a particular partition.

This is true for all partitions. Since no schedule lies in two partitions, and the partition cover $\mathbf{F}(S)$, we can conclude that the expected weight of the algorithm on the region of S_i must be at least $\frac{c_{\log \Delta}}{4}|S_i|$.

We then complete the proof.

Proving Theorem 5

Now it's ready to prove Theorem 5.

Note that no interval whose start point lies within S_i can be longer than $2|S_i|$, or it could be added to S_i. So any interval starting after S_i of length at least $4|S_i|$ would satisfy the conditions to be a valid start interval for S_{i+1}. Thus, the distance from the end of S_i to the beginning of S_{i+1} cannot be more than $6|S_i|$.

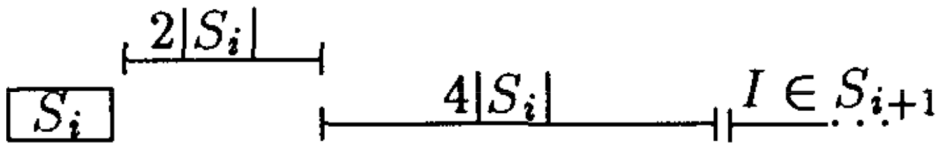


Figure 7: The space between S_i and S_{i+1} must be less than $6|S_i|$.

Thus, we have $\sum_i |S_i| \geq \frac{1}{7}|S|$.

Proving Theorem 5

Recall that we defined the region associated with an S_i as the set of intervals whose start points lie after the end of all intervals in S_{i-1} but before the end of the last interval of S_i. Call this region $R(S_i)$. We then have

$$\begin{aligned} W(\mathsf{MA}, S) &\geq \sum_i W_S(\mathsf{MA}, R(S_i)) \\ &\geq \sum_i \frac{c_{\log \Delta}}{4} |S_i| \\ &\geq \frac{c_{\log \Delta}}{4} \frac{|S|}{7} \\ &\geq \frac{c_{\log \Delta}}{4} \frac{|\sigma^*|}{7}. \end{aligned} \tag{21}$$

We then complete the proof of the theorem.

A Lower Bound

Consider the set σ_n consisting of intervals $I_0, ..., I_n$ and constructed as follows:

$$\forall k : \text{start}(I_k) = \frac{k}{n}, \text{and } |I_k| = 2^k. \tag{22}$$

All intervals in σ_n overlap, and they are ordered by length. The best choice is I_n.

For any algorithm as $n \to \infty$, we can think of the distribution $\{c_k\}$ over the positive integers, where

$$c_k = \Pr[\text{Algorithm takes } I_k].$$

Note that Δ in this case is 2^n, i.e., $n = \log \Delta$.

A Lower Bound

Theorem 7

There does not exist an algorithm which is $O(n)$-competitive on σ_n.

Now we present its proof.

Assume $\exists k \forall n \sum_{i=0}^{n} c_i 2^i \geq k\frac{2^n}{n}$ for some distribution $\{c_i\}_i$.
That is, assume there is an $O(n)$-competitive algorithm.

We consider the sequence $[c_1, ..., c_n]$ for some fixed n, and we take partial sums of $2\log n$ consecutive c_i's:

$$\sum_{i=d(2\log n)}^{(d+1)(2\log n)} c_i. \tag{23}$$

We shall show that any set of c_i's achieving the competitive bound will not form a valid distribution.

A Lower Bound

First, from the assumption, we have

$$\begin{aligned}\sum_{i=0}^{(d+1)(2\log n)} c_i 2^i &\geq k\frac{2^{(d+1)(2\log n)}}{(d+1)(2\log n)} \\ &= k\frac{(n^2)^{d+1}}{(d+1)(2\log n)} \\ &= (n^2)^d \frac{kn^2}{(d+1)(2\log n)} \qquad (24)\end{aligned}$$

and

$$\begin{aligned}\sum_{i=0}^{d(2\log n)} c_i 2^i &\leq 2^{d(2\log n)} \sum_{i=0}^{d(2\log n)} c_i \\ &\leq 2^{d(2\log n)} \\ &= (n^2)^d. \qquad (25)\end{aligned}$$

A Lower Bound

So there exists a $k' < k$ such that

$$\sum_{i=d(2\log n)}^{(d+1)(2\log n)} c_i \geq \frac{(n^2)^{d+1}}{(d+1)(2\log n)} \left(k - \frac{(d+1)2\log n}{n^2} \right)$$

$$= \frac{k'(n^2)^{d+1}}{(d+1)(2\log n)}. \tag{26}$$

We further have

$$2^{(d+1)(2\log n)} \sum_{i=d(2\log n)}^{(d+1)(2\log n)} c_i \geq \frac{k'(n^2)^{d+1}}{(d+1)(2\log n)}, \tag{27}$$

which leads to

$$\sum_{i=d(2\log n)}^{(d+1)(2\log n)} c_i \geq \frac{k'}{(d+1)(2\log n)}. \tag{28}$$

A Lower Bound

Note that k' can be chosen to be valid for all n and d beyond a certain n_0. For instance, $k' = k/2$ will have the property.

Summing all these partial series of c_i's over d ranging from 1 to $n/(2\log n)$, we get

$$\begin{aligned}\sum_{i=2\log n}^{n} c_i &\geq k'\left(\frac{1}{4\log n} + \frac{1}{6\log n} + \cdots + \frac{1}{n}\right) \\ &= \frac{k'}{2\log n}\left(\frac{1}{1} + \frac{1}{2} + \cdots + \frac{2\log n}{n} - 1\right) \\ &= \frac{k'}{2\log n}\left(\sum_{i=1}^{n/(2\log n)} \frac{1}{i} - 1\right) \qquad (29)\end{aligned}$$

When $n \to \infty$, the sum $\sum_{x=1}^{n} \frac{1}{x}$ (the harmonic series) diverges to infinity, while $\log_2 n$ also grows to infinity but at a much slower rate.

A Lower Bound

Thus, we further have

$$\begin{aligned}\text{RHS of 29} &\geq \frac{k'}{2\log n}\left(\log(\frac{n}{2\log n})-1\right)\\ &= \frac{k'}{2\log n}\left(\log n - \log\log n^2 - 1\right)\\ &= k'\left(\frac{1}{2}-\frac{\log\log n^2+1}{2\log n}\right)\\ &\geq \frac{k'}{4}. \end{aligned} \tag{30}$$

The last equation is because $\frac{\log\log n^2+1}{2\log n} \to 0$. It indicates that we have a lower bound for $\sum_{i=2\log n}^{n} c_i$.

A Lower Bound

We can then perform thje substitution $n \to \log n$, which wouold yield $\sum_{i=2\log\log n}^{\log n} c_i \geq k'/4$. Note that the c_i's from this new sum are disjoint from those of the previous sum. We can continue this process of taking logarithms until we reach a constant:

$$\sum_{i=1}^{n} c_i \geq \frac{k'}{4} \log^* n. \tag{31}$$

For any value of k', we can choose n large enough that $\log^* n > 4/k'$, yielding $\sum_i c_i > 1$. This contradicts that $\{c_i\}_i$ is a distribution.

We then have the lower bound proved.

Outline

Conclusion

For the online interval scheduling problem, we show the following results:

- A strongly 2-competitive algorithm for 2-length problem instances.
- An $O((\log \Delta)^{1+\epsilon})$-competitive algorithm for scheduling intervals of arbitrary lengths.
- An $O(\log \Delta)$ lower bound on the competitive ratio for scheduling intervals of arbitrary lengths.