Unit Interval Selection in Random Order Streams

Cezar-Mihail Alexandru[1], Adithya Diddapur[2], Magnús M. Halldórsson[3], Christian Konrad[2], Kheeran K. Naidu[4]

[1]Independent Researcher, [2]University of Bristol, [3]Reykjavik University, [4]Qworky Research

Interval Selection Problem

Interval Selection Problem

- An unit interval is a set $I = [a, a + 1] \subset \mathbb{R}$.
- I_1, I_2 are independent if $I_1 \cap I_2 = \emptyset$.

Interval Selection Problem

- An unit interval is a set $I = [a, a + 1] \subset \mathbb{R}$.
- I_1, I_2 are independent if $I_1 \cap I_2 = \emptyset$.

Problem (Unit Interval Selection)

Given a set of unit intervals, presented as a stream, return a set of pairwise independent intervals which is (approximately) as large as possible.

Interval Selection Problem

- An unit interval is a set $I = [a, a+1] \subset \mathbb{R}$.
- I_1, I_2 are independent if $I_1 \cap I_2 = \emptyset$.

Problem (Unit Interval Selection)

Given a set of unit intervals, presented as a stream, return a set of pairwise independent intervals which is (approximately) as large as possible.

Interval Selection Problem

- An unit interval is a set $I = [a, a+1] \subset \mathbb{R}$.
- I_1, I_2 are independent if $I_1 \cap I_2 = \emptyset$.

Problem (Unit Interval Selection)

Given a set of unit intervals, presented as a stream, return a set of pairwise independent intervals which is (approximately) as large as possible.

Interval Selection Problem

- An unit interval is a set $I = [a, a+1] \subset \mathbb{R}$.
- I_1, I_2 are independent if $I_1 \cap I_2 = \emptyset$.

Problem (Unit Interval Selection)

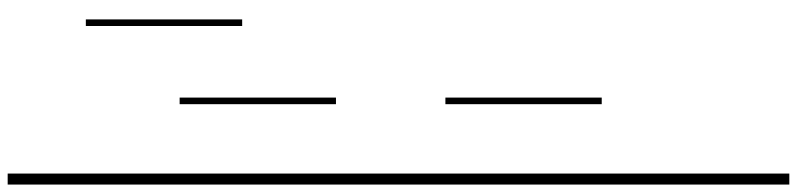
Given a set of unit intervals, presented as a stream, return a set of pairwise independent intervals which is (approximately) as large as possible.

Interval Selection Problem

- An unit interval is a set $I = [a, a+1] \subset \mathbb{R}$.
- I_1, I_2 are independent if $I_1 \cap I_2 = \emptyset$.

Problem (Unit Interval Selection)

Given a set of unit intervals, presented as a stream, return a set of pairwise independent intervals which is (approximately) as large as possible.

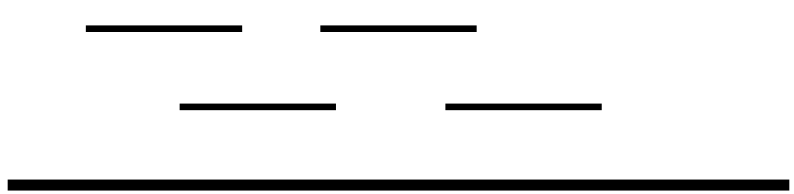


Interval Selection Problem

- An unit interval is a set $I = [a, a+1] \subset \mathbb{R}$.
- I_1, I_2 are independent if $I_1 \cap I_2 = \emptyset$.

Problem (Unit Interval Selection)

Given a set of unit intervals, presented as a stream, return a set of pairwise independent intervals which is (approximately) as large as possible.

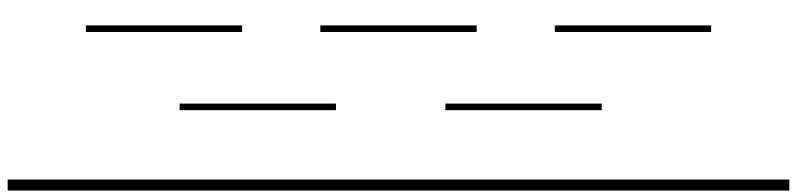


Interval Selection Problem

- An unit interval is a set $I = [a, a+1] \subset \mathbb{R}$.
- I_1, I_2 are independent if $I_1 \cap I_2 = \emptyset$.

Problem (Unit Interval Selection)

Given a set of unit intervals, presented as a stream, return a set of pairwise independent intervals which is (approximately) as large as possible.

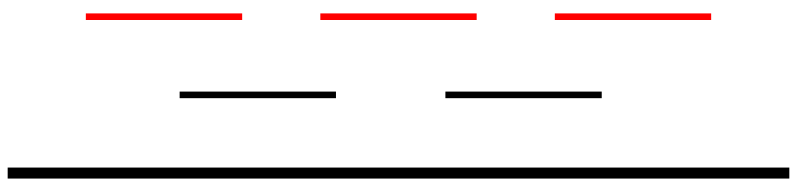


Interval Selection Problem

- An unit interval is a set $I = [a, a+1] \subset \mathbb{R}$.
- I_1, I_2 are independent if $I_1 \cap I_2 = \emptyset$.

Problem (Unit Interval Selection)

Given a set of unit intervals, presented as a stream, return a set of pairwise independent intervals which is (approximately) as large as possible.



The Streaming Setting

The Streaming Setting

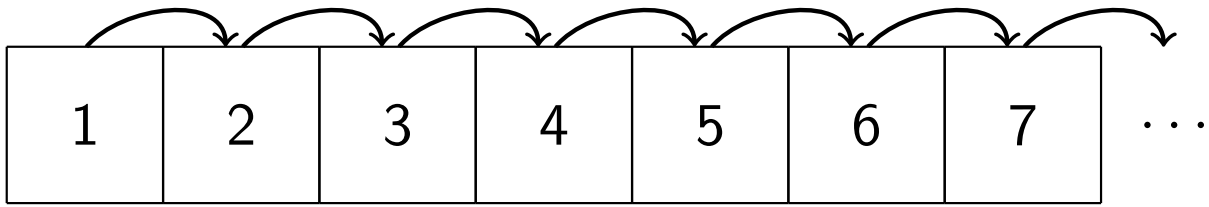
- We have an adversarial input set of intervals $\mathcal{I} = (I_1, \ldots, I_m)$.
- Unable to store the entire input.

The Streaming Setting

- We have an adversarial input set of intervals $\mathcal{I} = (I_1, \ldots, I_m)$.
- Unable to store the entire input.

Goal

Given the input elements one at a time, compute an (approximate) solution using as **little space** as possible.



The Streaming Setting (Contd.)

The Streaming Setting (Contd.)

- Length of the stream is by n.
- Optimum solution is denoted OPT.

The Streaming Setting (Contd.)

- Length of the stream is by n.
- Optimum solution is denoted OPT.

The problem can be solved in space $O(n)$ and requires $\Omega(|OPT|)$.

The Streaming Setting (Contd.)

- Length of the stream is by n.
- Optimum solution is denoted OPT.

The problem can be solved in space $O(n)$ and requires $\Omega(|OPT|)$.

Previous work studied when the stream is adversarially ordered.

The Streaming Setting (Contd.)

- Length of the stream is by n.
- Optimum solution is denoted OPT.

The problem can be solved in space $O(n)$ and requires $\Omega(|OPT|)$.

Previous work studied when the stream is adversarially ordered.

- A 2/3-approximation is the right answer [EHR12, CPL15].

Our Work

Our Work

We study the problem in random-order streams.

Our Work

We study the problem in random-order streams.

- Here, the input is still adversarially chosen, but then arrives in a uniform random order.

Our Work

We study the problem in random-order streams.

- Here, the input is still adversarially chosen, but then arrives in a uniform random order.

Our Work

We study the problem in random-order streams.

- Here, the input is still adversarially chosen, but then arrives in a uniform random order.

Theorem

There is a deterministic algorithm which uses space $O(|OPT|)$, and returns a 0.7401*-approximation in expectation.*

Our Work

We study the problem in random-order streams.

- Here, the input is still adversarially chosen, but then arrives in a uniform random order.

Theorem

There is a deterministic algorithm which uses space $O(|OPT|)$, and returns a 0.7401-approximation in expectation.

Theorem

1. *Any alg. which computes a better-than $8/9$-approximation in exp. must use space $\Omega(n)$.*
2. *Any alg. which computes a better than $2/3$-approximation w.p. $> 2/3$ must use space $\Omega(n)$.*

The Algorithm

Theorem

There is a deterministic algorithm which uses space $O(|OPT|)$, and returns a 0.7401*-approximation in expectation.*

The Algorithm

The Algorithm

- A window of length γ is a set of the form $[a, a + \gamma)$.

The Algorithm

- A window of length γ is a set of the form $[a, a + \gamma)$.

Lemma

If there is a unit-length algorithm for a restricted window of length γ, then there is an algorithm for unrestricted domains whose approximation factor decreases by $(\gamma - 1)/\gamma$ and space by $O(|OPT|)$.

The Algorithm

- A window of length γ is a set of the form $[a, a + \gamma)$.

Lemma

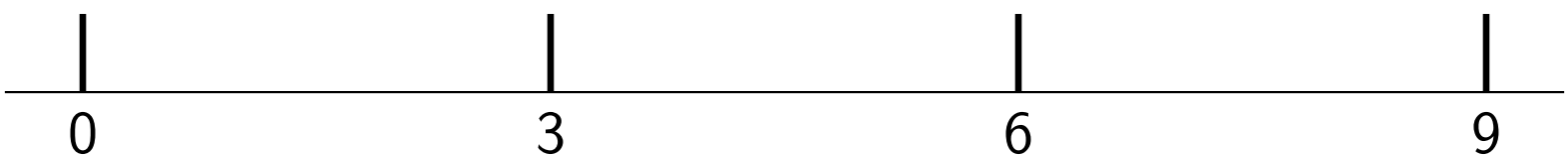
If there is a unit-length algorithm for a restricted window of length γ, then there is an algorithm for unrestricted domains whose approximation factor decreases by $(\gamma - 1)/\gamma$ and space by $O(|OPT|)$.

The Algorithm

- A window of length γ is a set of the form $[a, a + \gamma)$.

Lemma

If there is a unit-length algorithm for a restricted window of length γ, then there is an algorithm for unrestricted domains whose approximation factor decreases by $(\gamma - 1)/\gamma$ and space by $O(|OPT|)$.

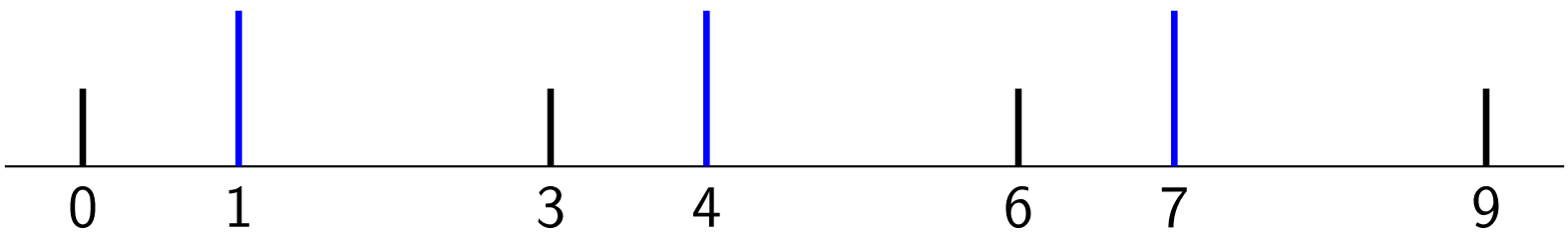


The Algorithm

- A window of length γ is a set of the form $[a, a + \gamma)$.

Lemma

If there is a unit-length algorithm for a restricted window of length γ, then there is an algorithm for unrestricted domains whose approximation factor decreases by $(\gamma - 1)/\gamma$ and space by $O(|OPT|)$.

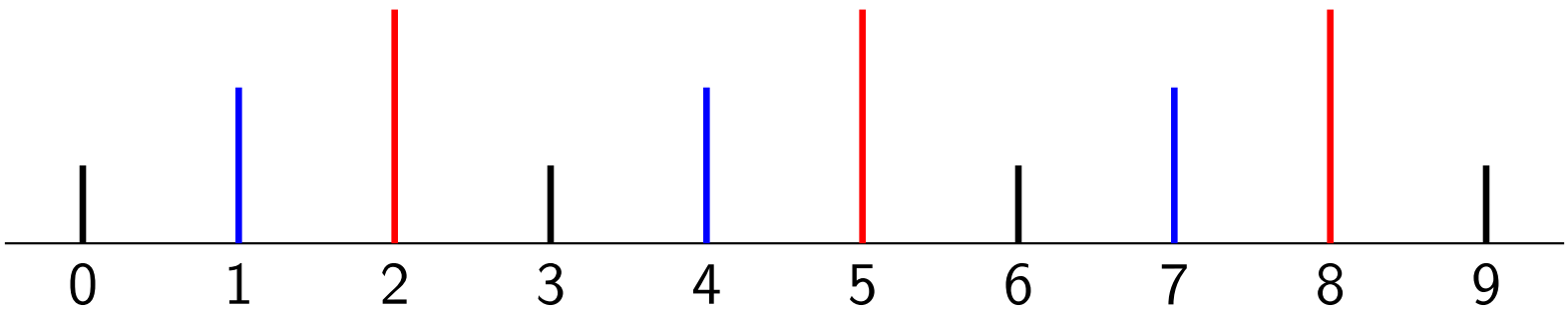


The Algorithm

- A window of length γ is a set of the form $[a, a + \gamma)$.

Lemma

If there is a unit-length algorithm for a restricted window of length γ, then there is an algorithm for unrestricted domains whose approximation factor decreases by $(\gamma - 1)/\gamma$ and space by $O(|OPT|)$.



The Algorithm

The Algorithm

- We now consider the restricted window $W = [0, \gamma)$ for some $\gamma > 0$.
- Let OPT_L denote an optimum inside this window.

The Algorithm

- We now consider the restricted window $W = [0, \gamma)$ for some $\gamma > 0$.
- Let OPT_L denote an optimum inside this window.

Observation

If the intervals in OPT_L arrive in order from left-to-right, then we can obtain a perfect solution.

The Algorithm

- We now consider the restricted window $W = [0, \gamma)$ for some $\gamma > 0$.
- Let OPT_L denote an optimum inside this window.

Observation

If the intervals in OPT_L arrive in order from left-to-right, then we can obtain a perfect solution.

- This is unlikely, but what if either the left or right-most arrives first?

The Algorithm

The Algorithm

This gives us an approach:

The Algorithm

This gives us an approach:

1. Store the left and right-most intervals at any time.

The Algorithm

This gives us an approach:

1. Store the left and right-most intervals at any time.
2. Maintain recursive instances which are fed everything further right or left respectively.

The Algorithm

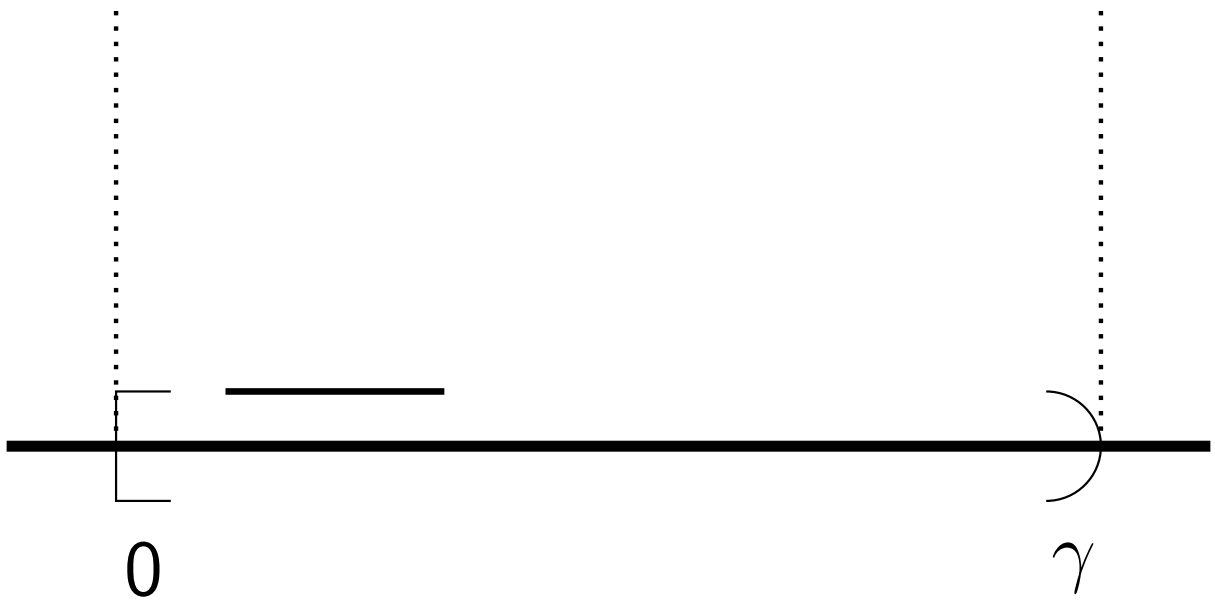
This gives us an approach:

1. Store the left and right-most intervals at any time.
2. Maintain recursive instances which are fed everything further right or left respectively.

The Algorithm

This gives us an approach:

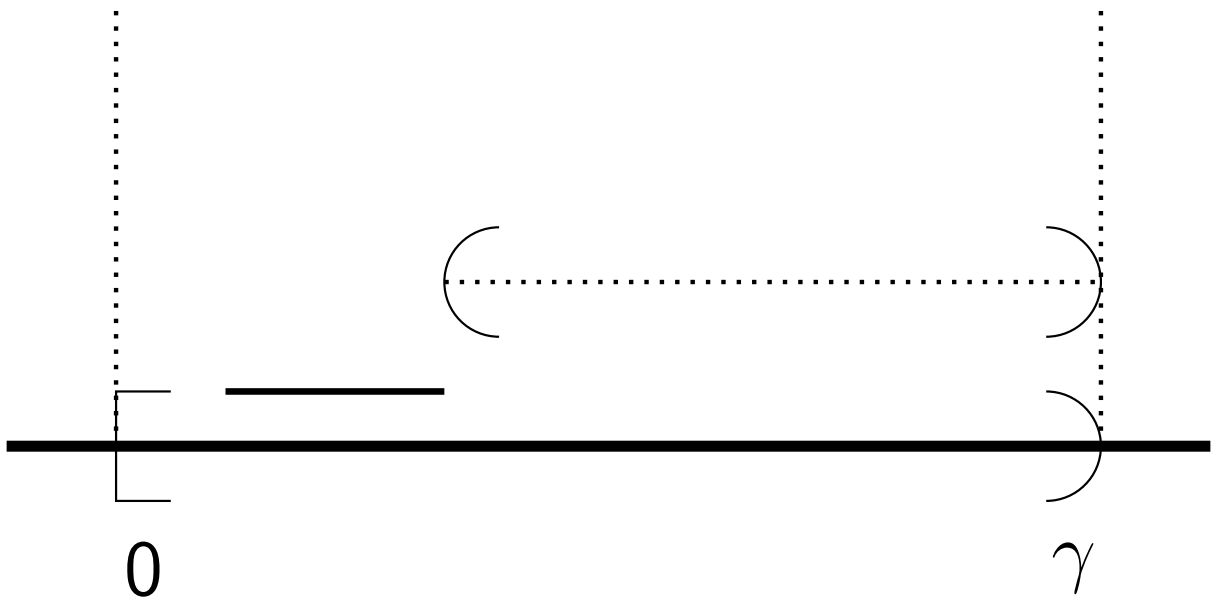
1. Store the left and right-most intervals at any time.
2. Maintain recursive instances which are fed everything further right or left respectively.



The Algorithm

This gives us an approach:

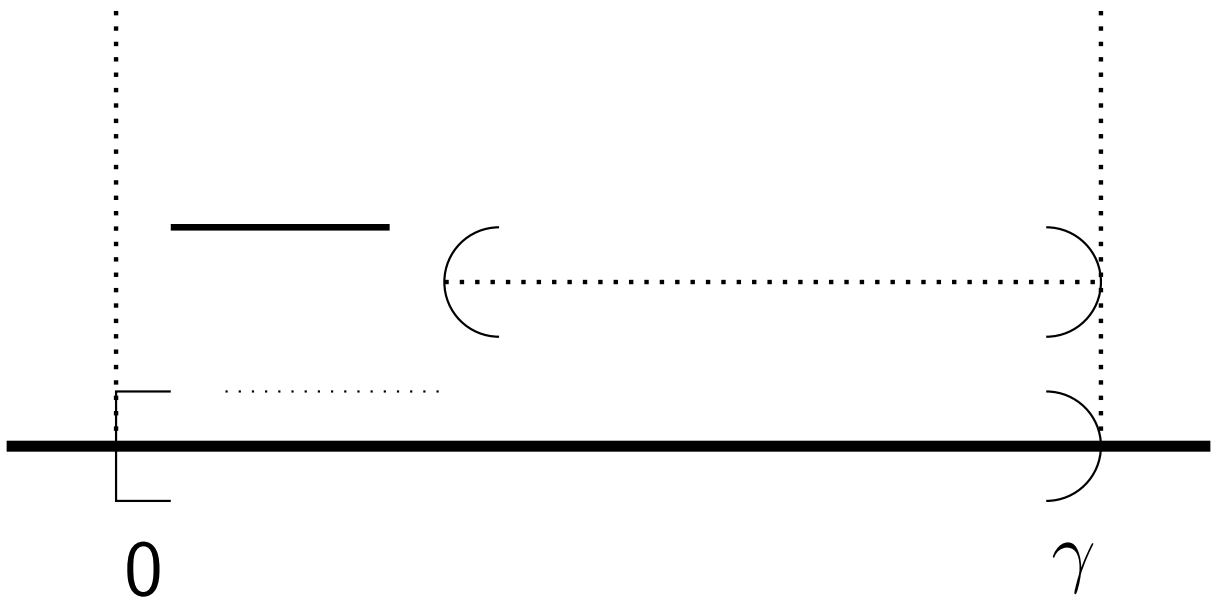
1. Store the left and right-most intervals at any time.
2. Maintain recursive instances which are fed everything further right or left respectively.



The Algorithm

This gives us an approach:

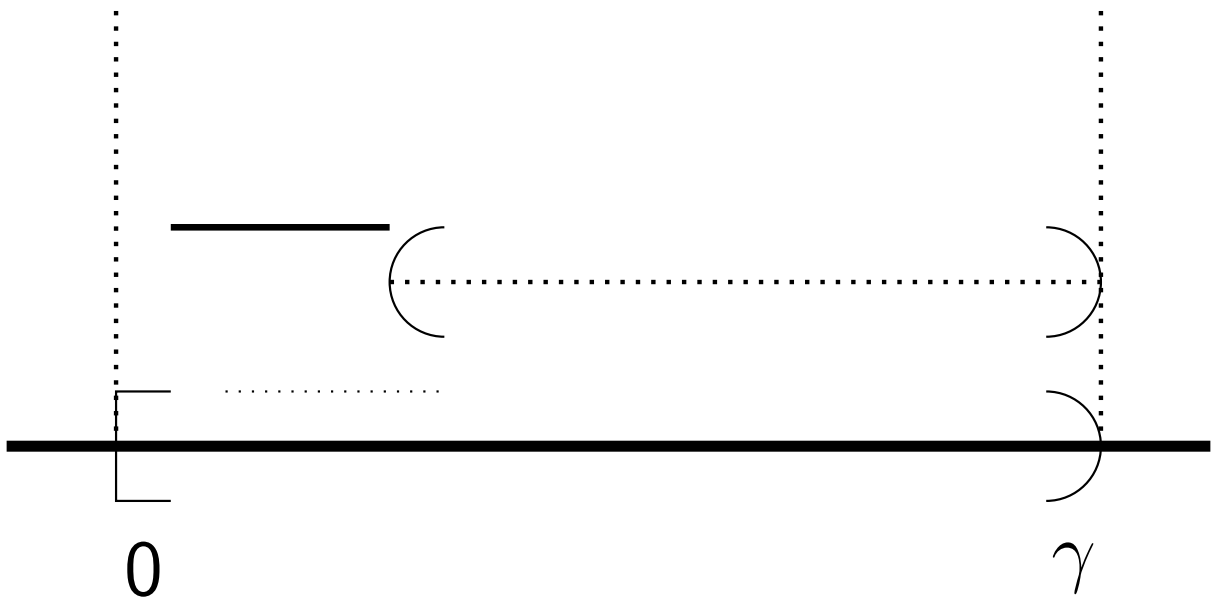
1. Store the left and right-most intervals at any time.
2. Maintain recursive instances which are fed everything further right or left respectively.



The Algorithm

This gives us an approach:

1. Store the left and right-most intervals at any time.
2. Maintain recursive instances which are fed everything further right or left respectively.



The Algorithm

The Algorithm

However, it is still unlikely for the left or right-most to arrive first.

The Algorithm

However, it is still unlikely for the left or right-most to arrive first.

- We overcome this by making lots of simultaneous guesses.

The Algorithm

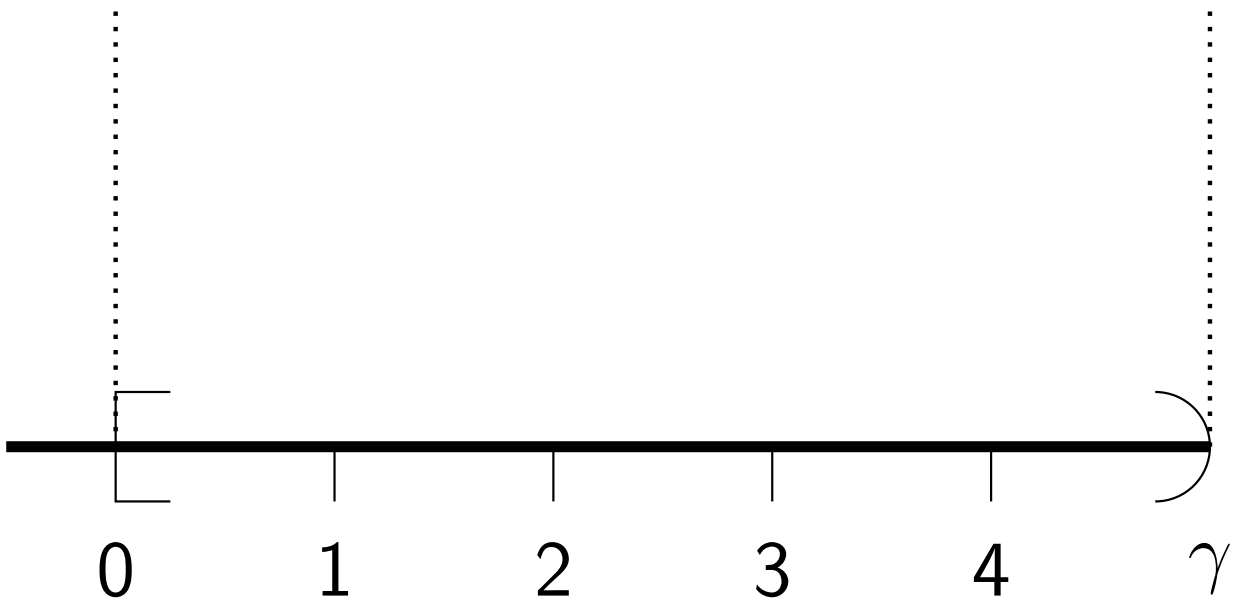
However, it is still unlikely for the left or right-most to arrive first.

- We overcome this by making lots of simultaneous guesses.
- For each integer $i \in [\gamma]$, we create a pair of instances around i.

The Algorithm

However, it is still unlikely for the left or right-most to arrive first.

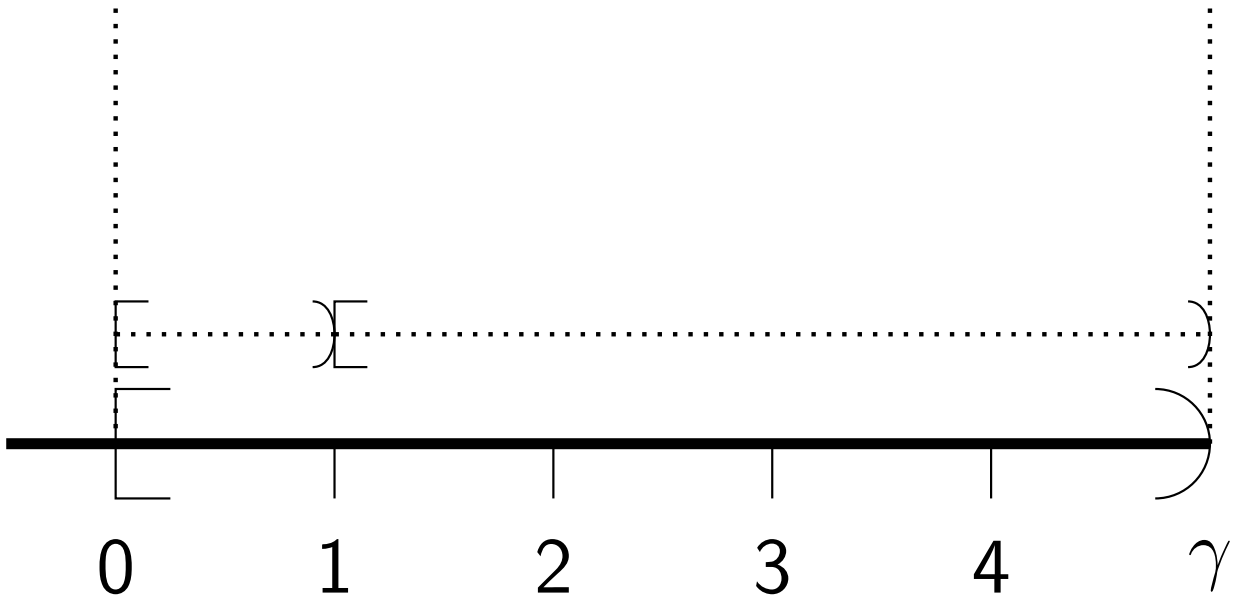
- We overcome this by making lots of simultaneous guesses.
- For each integer $i \in [\gamma]$, we create a pair of instances around i.



The Algorithm

However, it is still unlikely for the left or right-most to arrive first.

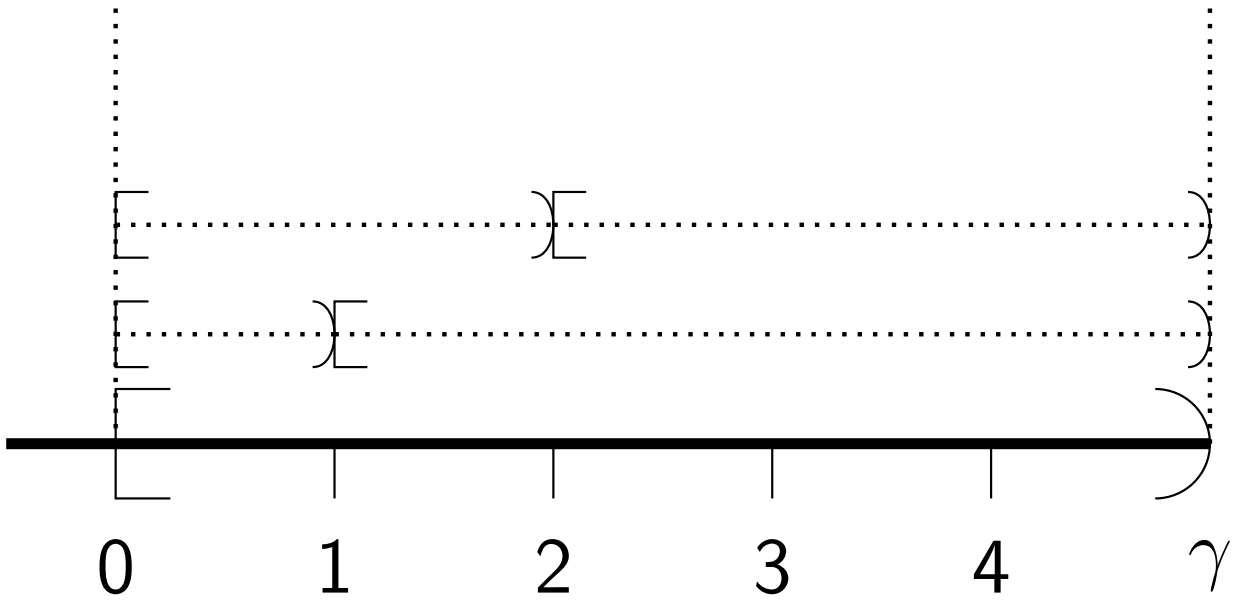
- We overcome this by making lots of simultaneous guesses.
- For each integer $i \in [\gamma]$, we create a pair of instances around i.



The Algorithm

However, it is still unlikely for the left or right-most to arrive first.

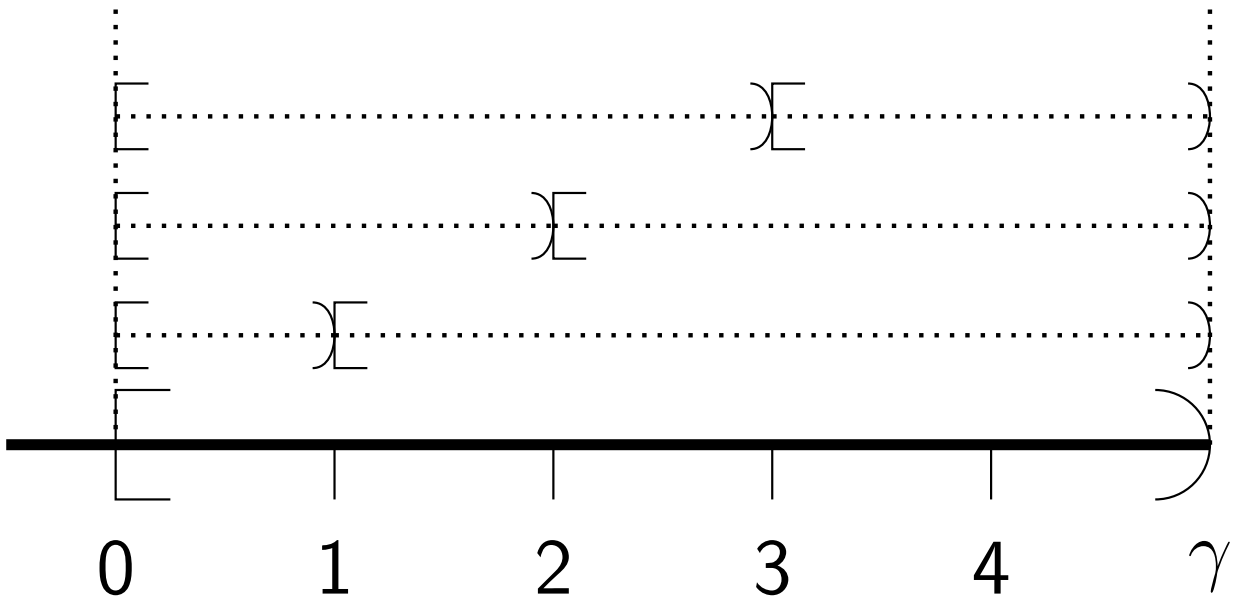
- We overcome this by making lots of simultaneous guesses.
- For each integer $i \in [\gamma]$, we create a pair of instances around i.



The Algorithm

However, it is still unlikely for the left or right-most to arrive first.

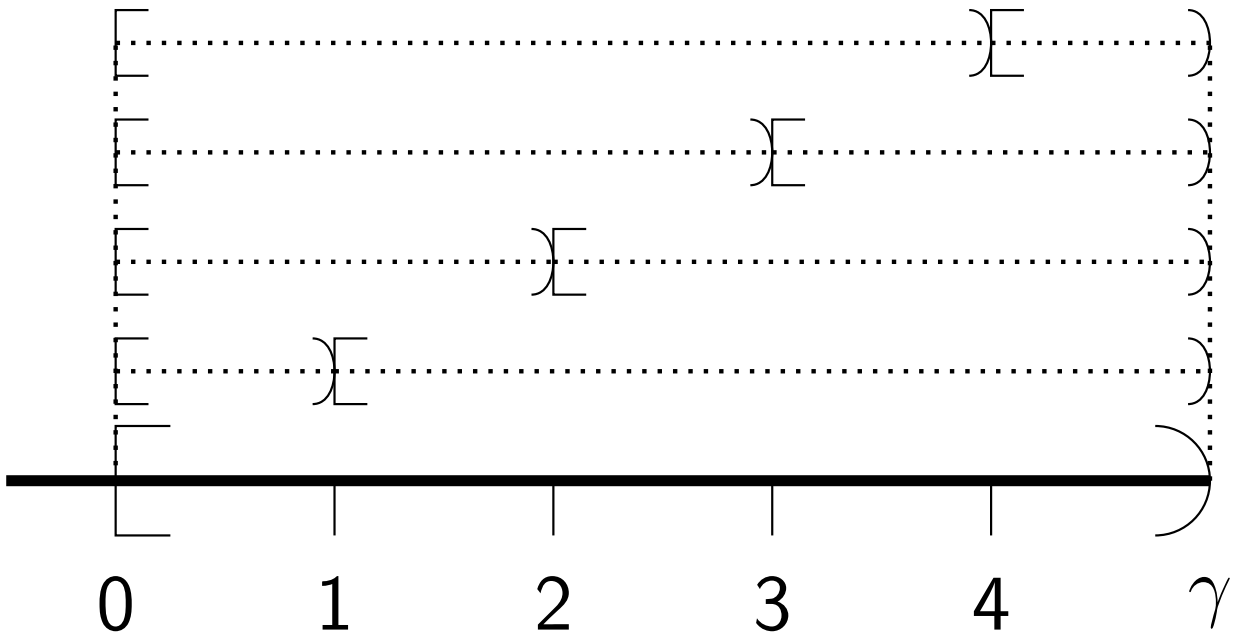
- We overcome this by making lots of simultaneous guesses.
- For each integer $i \in [\gamma]$, we create a pair of instances around i.



The Algorithm

However, it is still unlikely for the left or right-most to arrive first.

- We overcome this by making lots of simultaneous guesses.
- For each integer $i \in [\gamma]$, we create a pair of instances around i.



The Algorithm

The Algorithm

Our analysis then proceeds as follows:

The Algorithm

Our analysis then proceeds as follows:

1. We obtain a recurrence relation for the expected output size, as a function of $|OPT_L|$.

The Algorithm

Our analysis then proceeds as follows:

1. We obtain a recurrence relation for the expected output size, as a function of $|OPT_L|$.
2. This yields another recurrence for the expected approximation factor.

The Algorithm

Our analysis then proceeds as follows:

1. We obtain a recurrence relation for the expected output size, as a function of $|OPT_L|$.
2. This yields another recurrence for the expected approximation factor.
3. We then incorporate the $\frac{\gamma-1}{\gamma}$ penalty from the shifting window technique to obtain another recurrence.

The Algorithm

Our analysis then proceeds as follows:

1. We obtain a recurrence relation for the expected output size, as a function of $|OPT_L|$.
2. This yields another recurrence for the expected approximation factor.
3. We then incorporate the $\frac{\gamma-1}{\gamma}$ penalty from the shifting window technique to obtain another recurrence.
4. Finally, we optimise this numerically.

The Algorithm

Our analysis then proceeds as follows:

1. We obtain a recurrence relation for the expected output size, as a function of $|OPT_L|$.
2. This yields another recurrence for the expected approximation factor.
3. We then incorporate the $\frac{\gamma-1}{\gamma}$ penalty from the shifting window technique to obtain another recurrence.
4. Finally, we optimise this numerically.
5. $\gamma = 5000$ yields a 0.7401-approximation.

The Lower Bound

The Lower Bound

Theorem

1. *Any alg. which computes a better-than* $8/9$*-approximation in exp. must use space* $\Omega(n)$.
2. *Any alg. which computes a better than* $2/3$*-approximation w.p.* $> 2/3$ *must use space* $\Omega(n)$.

The Lower Bound

The Lower Bound

- We proceed via one-way two-party communication complexity.
- Our approach is similar to the robust communication complexity arguments of [CCM08].

The Lower Bound

- We proceed via one-way two-party communication complexity.
- Our approach is similar to the robust communication complexity arguments of [CCM08].

INDEX Problem - INDEX_t

Alice	Bob
$X \in \{0,1\}^t$	$\sigma \in [t]$

The Lower Bound

- We proceed via one-way two-party communication complexity.
- Our approach is similar to the robust communication complexity arguments of [CCM08].

INDEX Problem - INDEX_t

$$\underline{\text{Alice}} \xrightarrow{\quad M \quad} \underline{\text{Bob}}$$

$$X \in \{0,1\}^t \qquad\qquad \sigma \in [t]$$

The Lower Bound

- We proceed via one-way two-party communication complexity.
- Our approach is similar to the robust communication complexity arguments of [CCM08].

INDEX Problem - INDEX_t

$$\underline{\text{Alice}} \xrightarrow{\quad M \quad} \underline{\text{Bob}}$$

$$X \in \{0,1\}^t \qquad\qquad \sigma \in [t]$$

- **Goal:** Output $X[\sigma]$.

The Lower Bound

- We proceed via one-way two-party communication complexity.
- Our approach is similar to the robust communication complexity arguments of [CCM08].

INDEX Problem - INDEX_t

$$\underline{\text{Alice}} \xrightarrow{\quad M \quad} \underline{\text{Bob}}$$

$$X \in \{0,1\}^t \qquad\qquad \sigma \in [t]$$

- **Goal:** Output $X[\sigma]$.

Theorem

[RY20] Solving INDEX_t requires a message of size $\Omega(t)$.

The Lower Bound

INDEX_t

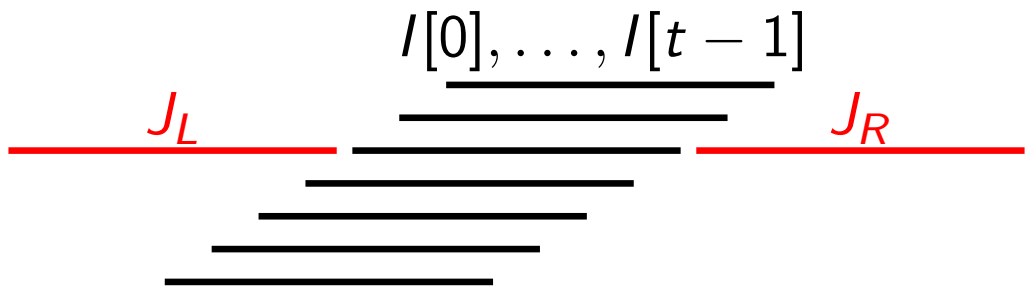
1. Alice holds $X \in \{0,1\}^t$, Bob holds $\sigma \in [t]$.
2. **Goal:** Output $X[\sigma]$.

The Lower Bound

INDEX_t

1. Alice holds $X \in \{0,1\}^t$, Bob holds $\sigma \in [t]$.
2. **Goal:** Output $X[\sigma]$.

The players construct a set of intervals which look like this, and do so via a shared uniform random ordering:

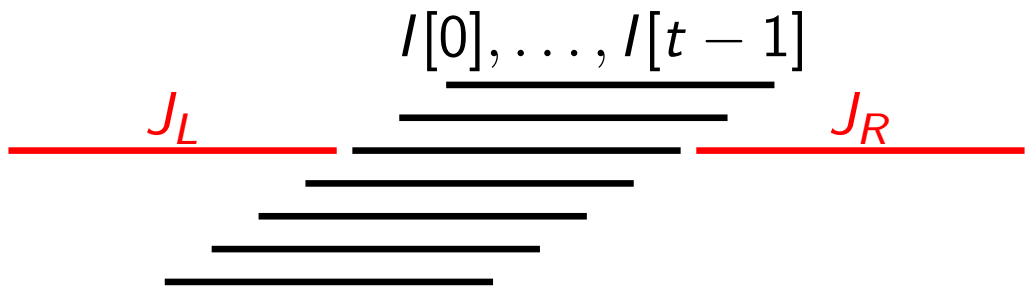


The Lower Bound

INDEX_t

1. Alice holds $X \in \{0,1\}^t$, Bob holds $\sigma \in [t]$.
2. **Goal:** Output $X[\sigma]$.

The players construct a set of intervals which look like this, and do so via a shared uniform random ordering:



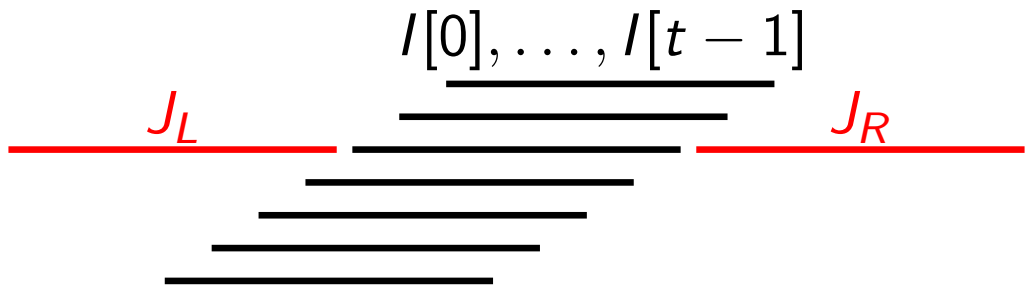
- The unique optimal solution of size three reveals the INDEX_t answer precisely when $I[\sigma]$ precedes both J_L and J_R.

The Lower Bound

INDEX_t

1. Alice holds $X \in \{0,1\}^t$, Bob holds $\sigma \in [t]$.
2. **Goal:** Output $X[\sigma]$.

The players construct a set of intervals which look like this, and do so via a shared uniform random ordering:



- The unique optimal solution of size three reveals the INDEX_t answer precisely when $I[\sigma]$ precedes both J_L and J_R.
- This event occurs with probability $2/3$.

Further Questions

We have shown a strict separation with adversarial-order unit-length interval selection.

Further Questions

We have shown a strict separation with adversarial-order unit-length interval selection.

- Can we further close this gap between 0.7401 and $8/9 \sim 0.889$?

Further Questions

We have shown a strict separation with adversarial-order unit-length interval selection.

- Can we further close this gap between 0.7401 and $8/9 \sim 0.889$?
- Can we beat the adversarial-order bounds known for variable-length intervals?

References I

Amit Chakrabarti, Graham Cormode, and Andrew McGregor.
Robust lower bounds for communication and stream computation.
In Proceedings of the fortieth annual ACM symposium on Theory of computing, pages 641–650, 2008.

Sergio Cabello and Pablo Pérez-Lantero.
Interval selection in the streaming model.
In Frank Dehne, Jörg-Rüdiger Sack, and Ulrike Stege, editors, Algorithms and Data Structures, pages 127–139, Cham, 2015. Springer International Publishing.

Yuval Emek, Magnus M. Halldorsson, and Adi Rosen.
Space-Constrained Interval Selection.
In
39th International Colloquium on Automata, Languages, and Prograr
July 2012.

References II

Anup Rao and Amir Yehudayoff.
Communication Complexity: and Applications.
Cambridge University Press, 2020.