Interval Selection in Insertion-Only Streams

Jacques Dark, Adithya Diddapur[1], Christian Konrad[1]

[1]University of Bristol

The Streaming Setting

The Streaming Setting

- We have an input $\mathcal{I} = (I_1, \ldots, I_m)$.

The Streaming Setting

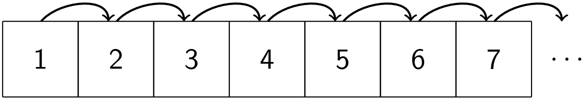
- We have an input $\mathcal{I} = (I_1, \ldots, I_m)$.
- Unable to store the entire input.

The Streaming Setting

- We have an input $\mathcal{I} = (I_1, \ldots, I_m)$.
- 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.)

- **Insertion-only**: stream elements arrive, and never disappear.

The Streaming Setting (Contd.)

- **Insertion-only**: stream elements arrive, and never disappear.
- **Insertion-deletion:** stream elements may either be elements arriving, or previous elements begin deleted.

The Streaming Setting (Contd.)

- **Insertion-only**: stream elements arrive, and never disappear.
- **Insertion-deletion:** stream elements may either be elements arriving, or previous elements begin deleted.

The Streaming Setting (Contd.)

- **Insertion-only**: stream elements arrive, and never disappear.
- **Insertion-deletion:** stream elements may either be elements arriving, or previous elements begin deleted.

- Length of the stream is by n.

The Streaming Setting (Contd.)

- **Insertion-only**: stream elements arrive, and never disappear.
- **Insertion-deletion:** stream elements may either be elements arriving, or previous elements begin deleted.

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

The Streaming Setting (Contd.)

- **Insertion-only**: stream elements arrive, and never disappear.
- **Insertion-deletion:** stream elements may either be elements arriving, or previous elements begin deleted.

- Length of the stream is by n.
- Optimum solution is denoted OPT.
- OPT^* denotes the largest solution over all prefixes.

The Streaming Setting (Contd.)

- **Insertion-only**: stream elements arrive, and never disappear.
- **Insertion-deletion:** stream elements may either be elements arriving, or previous elements begin deleted.

- Length of the stream is by n.
- Optimum solution is denoted OPT.
- OPT^* denotes the largest solution over all prefixes.

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

The Streaming Setting (Contd.)

- **Insertion-only**: stream elements arrive, and never disappear.
- **Insertion-deletion:** stream elements may either be elements arriving, or previous elements begin deleted.

- Length of the stream is by n.
- Optimum solution is denoted OPT.
- OPT^* denotes the largest solution over all prefixes.

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

- **Key Question:** What can we achieve across this regime?

Interval Selection Problem

Interval Selection Problem

- An interval is a set $I = [a, b] \subset \mathbb{R}$.

Interval Selection Problem

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

Interval Selection Problem

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

Problem (Interval Selection)

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

Interval Selection Problem

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

Problem (Interval Selection)

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

Interval Selection Problem

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

Problem (Interval Selection)

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

Interval Selection Problem

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

Problem (Interval Selection)

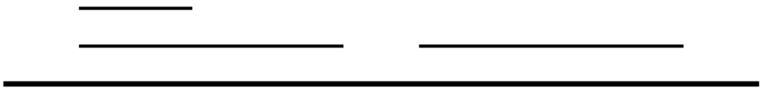
Given a set of intervals, presented as a stream, return a set of pairwise independent intervals which is (approximately) as large as possible.

Interval Selection Problem

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

Problem (Interval Selection)

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

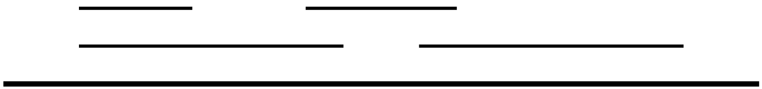


Interval Selection Problem

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

Problem (Interval Selection)

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

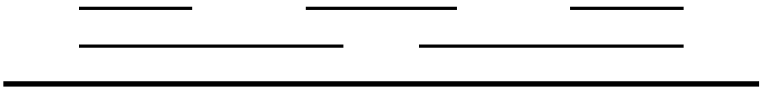


Interval Selection Problem

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

Problem (Interval Selection)

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

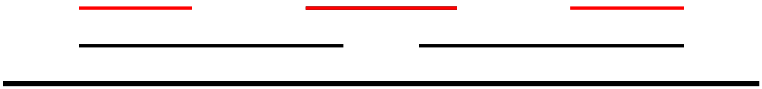


Interval Selection Problem

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

Problem (Interval Selection)

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



State of the Art

		Insertion-Only		Insertion-Deletion	
		UB - $O_\epsilon(\lvert OPT \rvert)$	LB - $\Omega(n)$	UB - $\tilde{O}_\epsilon(\lvert OPT^* \rvert)$	LB - $\Omega(n)$
Unweighted	Unit-Length				
	Variable-Length				
Weighted	Unit-Length				
	Variable-Length				

State of the Art

		Insertion-Only		Insertion-Deletion	
		UB - $O_\epsilon(\lvert OPT \rvert)$	LB - $\Omega(n)$	UB - $\tilde{O}_\epsilon(\lvert OPT^* \rvert)$	LB - $\Omega(n)$
Unweighted	Unit-Length	$3/2$ [EHR12]	$3/2-\epsilon$ [EHR12]		$2-\epsilon$ [BCW20]
	Variable-Length	2 [EHR12]	$2-\epsilon$ [EHR12]		
Weighted	Unit-Length		$3/2-\epsilon$ [EHR12]		$2-\epsilon$ [BCW20]
	Variable-Length				

State of the Art

		Insertion-Only		Insertion-Deletion	
		UB - $O_\epsilon(\lvert OPT \rvert)$	LB - $\Omega(n)$	UB - $\tilde{O}_\epsilon(\lvert OPT^* \rvert)$	LB - $\Omega(n)$
Unweighted	Unit-Length	**$3/2$** [EHR12]	$3/2 - \epsilon$ [EHR12]		$2 - \epsilon$ [BCW20]
	Variable-Length	2 [EHR12]	$2 - \epsilon$ [EHR12]		
Weighted	Unit-Length		$3/2 - \epsilon$ [EHR12]		$2 - \epsilon$ [BCW20]
	Variable-Length				

State of the Art

		Insertion-Only		Insertion-Deletion	
		UB - $O_\epsilon(\lvert OPT\rvert)$	LB - $\Omega(n)$	UB - $\tilde{O}_\epsilon(\lvert OPT^*\rvert)$	LB - $\Omega(n)$
Unweighted	Unit-Length	$3/2$ [EHR12]	$3/2-\epsilon$ [EHR12]		$2-\epsilon$ [BCW20]
	Variable-Length	2 [EHR12]	$2-\epsilon$ [EHR12]		
Weighted	Unit-Length	$\mathbf{3/2}+\epsilon$	$3/2-\epsilon$ [EHR12]		$2-\epsilon$ [BCW20]
	Variable-Length				

State of the Art

		Insertion-Only		Insertion-Deletion	
		UB - $O_\epsilon(\lvert OPT \rvert)$	LB - $\Omega(n)$	UB - $\tilde{O}_\epsilon(\lvert OPT^* \rvert)$	LB - $\Omega(n)$
Unweighted	Unit-Length	$3/2$ [EHR12]	$\mathbf{3/2} - \epsilon$ [EHR12]		$2 - \epsilon$ [BCW20]
	Variable-Length	2 [EHR12]	$2 - \epsilon$ [EHR12]		
Weighted	Unit-Length	$\mathbf{3/2} + \epsilon$	$3/2 - \epsilon$ [EHR12]		$2 - \epsilon$ [BCW20]
	Variable-Length				

State of the Art

		Insertion-Only		Insertion-Deletion	
		UB - $O_\epsilon(\lvert OPT\rvert)$	LB - $\Omega(n)$	UB - $\tilde{O}_\epsilon(\lvert OPT^*\rvert)$	LB - $\Omega(n)$
Unweighted	Unit-Length	$3/2$ [EHR12]	$3/2-\epsilon$ [EHR12]		$2-\epsilon$ [BCW20]
	Variable-Length	2 [EHR12]	$2-\epsilon$ [EHR12]		
Weighted	Unit-Length	$\mathbf{3/2}+\epsilon$	$3/2-\epsilon$ [EHR12]		$2-\epsilon$ [BCW20]
	Variable-Length				

State of the Art

		Insertion-Only		Insertion-Deletion	
		UB - $O_\epsilon(\|OPT\|)$	LB - $\Omega(n)$	UB - $\tilde{O}_\epsilon(\|OPT^*\|)$	LB - $\Omega(n)$
Unweighted	Unit-Length	$3/2$ [EHR12]	$3/2-\epsilon$ [EHR12]	$\mathbf{2}$	$2-\epsilon$ [BCW20]
	Variable-Length	2 [EHR12]	$2-\epsilon$ [EHR12]	$*$	$\mathbf{\Theta(1)}$
Weighted	Unit-Length	$\mathbf{3/2}+\epsilon$	$3/2-\epsilon$ [EHR12]	$\mathbf{2}+\epsilon$	$2-\epsilon$ [BCW20]
	Variable-Length	$*$	$\mathbf{\Theta(1)}$	$*$	$\mathbf{\Theta(1)}$

Unit-Length Intervals

Unit-Length Intervals

- **Key Idea:** Let's break this down into a simpler problem.

Unit-Length Intervals

- **Key Idea:** Let's break this down into a simpler problem.

Lemma

[CPL17] If there is a unit-length algorithm for a restricted window of length 3, then there is an unrestricted algorithm which computes a 3/2-approximation.

Unit-Length Intervals

- **Key Idea:** Let's break this down into a simpler problem.

Lemma

[CPL17] If there is a unit-length algorithm for a restricted window of length 3*, then there is an unrestricted algorithm which computes a* $3/2$*-approximation.*

Unit-Length Intervals

- **Key Idea:** Let's break this down into a simpler problem.

Lemma

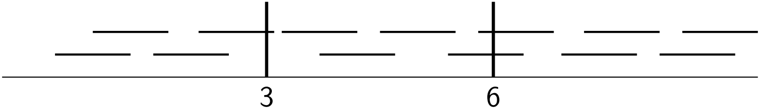
[CPL17] If there is a unit-length algorithm for a restricted window of length 3, then there is an unrestricted algorithm which computes a 3/2-approximation.

Unit-Length Intervals

- **Key Idea:** Let's break this down into a simpler problem.

Lemma

[CPL17] If there is a unit-length algorithm for a restricted window of length 3*, then there is an unrestricted algorithm which computes a* $3/2$*-approximation.*

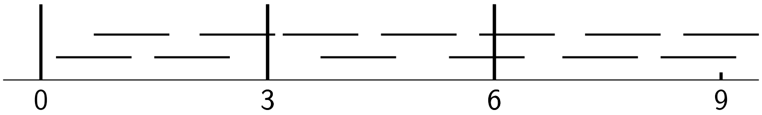


Unit-Length Intervals

- **Key Idea:** Let's break this down into a simpler problem.

Lemma

[CPL17] If there is a unit-length algorithm for a restricted window of length 3, then there is an unrestricted algorithm which computes a 3/2-approximation.

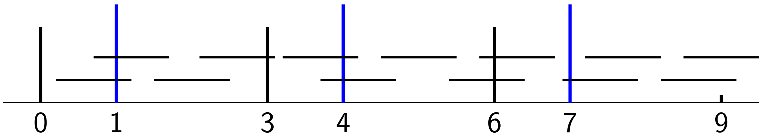


Unit-Length Intervals

- **Key Idea:** Let's break this down into a simpler problem.

Lemma

[CPL17] If there is a unit-length algorithm for a restricted window of length 3, then there is an unrestricted algorithm which computes a 3/2-approximation.



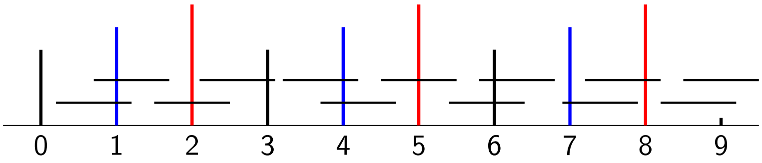
- If we can solve the problem for windows of length 3, then we get a 3/2-approximation.

Unit-Length Intervals

- **Key Idea:** Let's break this down into a simpler problem.

Lemma

[CPL17] If there is a unit-length algorithm for a restricted window of length 3, then there is an unrestricted algorithm which computes a 3/2-approximation.



- If we can solve the problem for windows of length 3, then we get a 3/2-approximation.

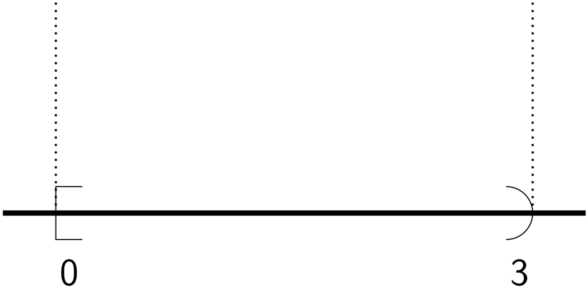
Unweighted Unit-Length Intervals

Unweighted Unit-Length Intervals

- Now consider a single window of length 3.

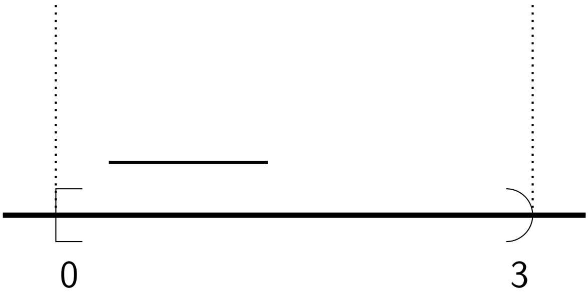
Unweighted Unit-Length Intervals

- Now consider a single window of length 3.



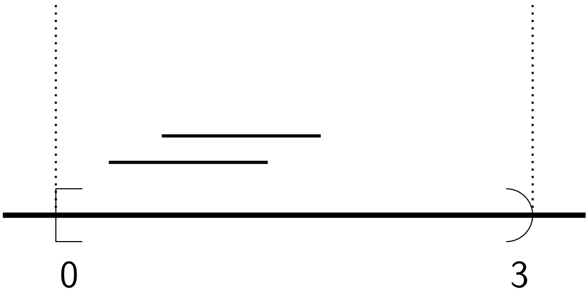
Unweighted Unit-Length Intervals

- Now consider a single window of length 3.



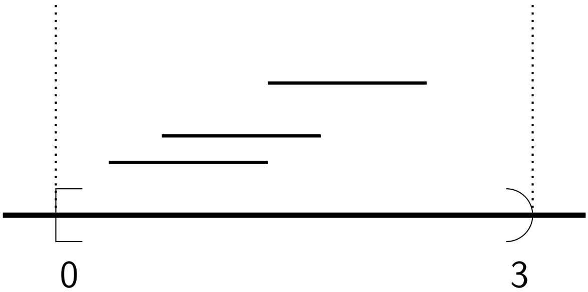
Unweighted Unit-Length Intervals

- Now consider a single window of length 3.



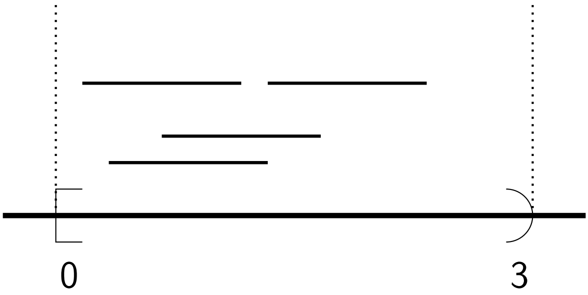
Unweighted Unit-Length Intervals

- Now consider a single window of length 3.



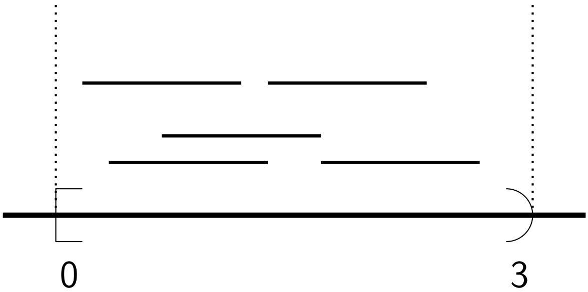
Unweighted Unit-Length Intervals

- Now consider a single window of length 3.



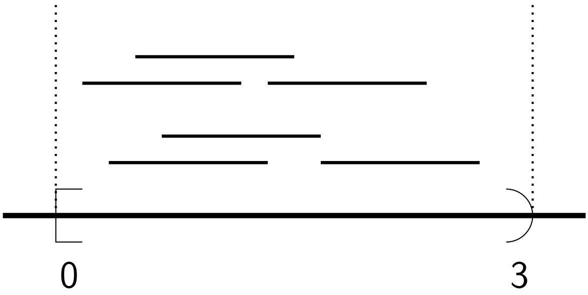
Unweighted Unit-Length Intervals

- Now consider a single window of length 3.



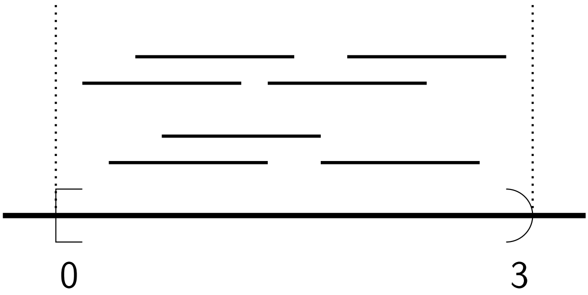
Unweighted Unit-Length Intervals

- Now consider a single window of length 3.



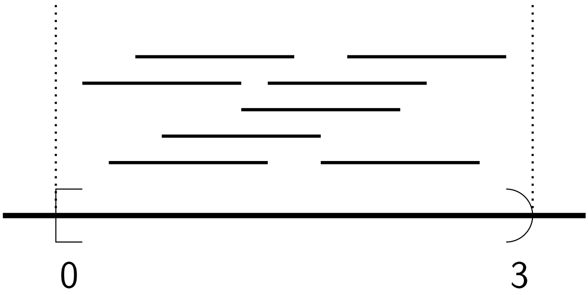
Unweighted Unit-Length Intervals

- Now consider a single window of length 3.



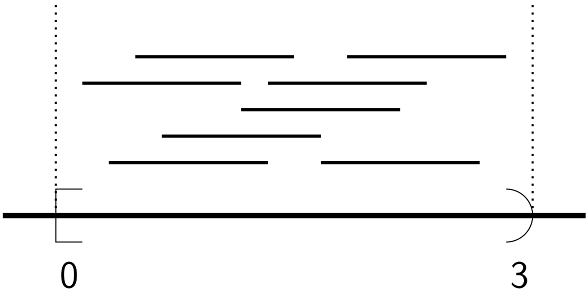
Unweighted Unit-Length Intervals

- Now consider a single window of length 3.



Unweighted Unit-Length Intervals

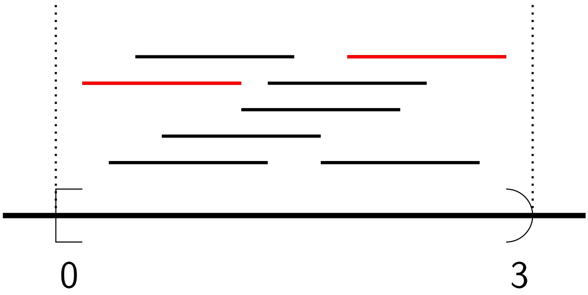
- Now consider a single window of length 3.



- **Unweighted Setting:** Find the left and right most intervals.

Unweighted Unit-Length Intervals

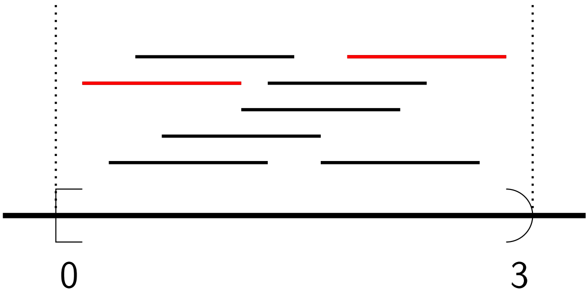
- Now consider a single window of length 3.



- **Unweighted Setting:** Find the left and right most intervals.

Unweighted Unit-Length Intervals

- Now consider a single window of length 3.



- **Unweighted Setting:** Find the left and right most intervals.
- This gives a 3/2-approximation by our earlier argument.

State of the Art

		Insertion-Only		Insertion-Deletion	
		UB - $O_\epsilon(\|OPT\|)$	LB - $\Omega(n)$	UB - $\tilde{O}_\epsilon(\|OPT^*\|)$	LB - $\Omega(n)$
Unweighted	Unit-Length	$\mathbf{3/2}$ [EHR12]	$3/2-\epsilon$ [EHR12]	$\mathbf{2}$	$2-\epsilon$ [BCW20]
	Variable-Length	2 [EHR12]	$2-\epsilon$ [EHR12]	$*$	$\mathbf{\Theta(1)}$
Weighted	Unit-Length	$3/2+\epsilon$	$3/2-\epsilon$ [EHR12]	$\mathbf{2}+\epsilon$	$2-\epsilon$ [BCW20]
	Variable-Length	$*$	$\mathbf{\Theta(1)}$	$*$	$\mathbf{\Theta(1)}$

State of the Art

		Insertion-Only		Insertion-Deletion	
		UB - $O_\epsilon(\lvert OPT \rvert)$	LB - $\Omega(n)$	UB - $\tilde{O}_\epsilon(\lvert OPT^* \rvert)$	LB - $\Omega(n)$
Unweighted	Unit-Length	$3/2$ [EHR12]	$3/2 - \epsilon$ [EHR12]	$\mathbf{2}$	$2 - \epsilon$ [BCW20]
	Variable-Length	2 [EHR12]	$2 - \epsilon$ [EHR12]	$*$	$\boldsymbol{\Theta(1)}$
Weighted	Unit-Length	$\mathbf{3/2} + \epsilon$	$3/2 - \epsilon$ [EHR12]	$\mathbf{2} + \epsilon$	$2 - \epsilon$ [BCW20]
	Variable-Length	$*$	$\boldsymbol{\Theta(1)}$	$*$	$\boldsymbol{\Theta(1)}$

Weighted Unit-Length Intervals

Weighted Unit-Length Intervals

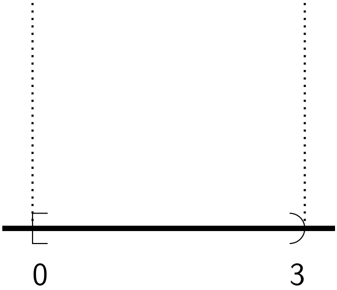
- **Weighted Setting:** Intervals have weights, and we want to pick heavy intervals.

Weighted Unit-Length Intervals

- **Weighted Setting:** Intervals have weights, and we want to pick heavy intervals.
- **Key Idea:** Sub-divide the window into geometric weight classes.

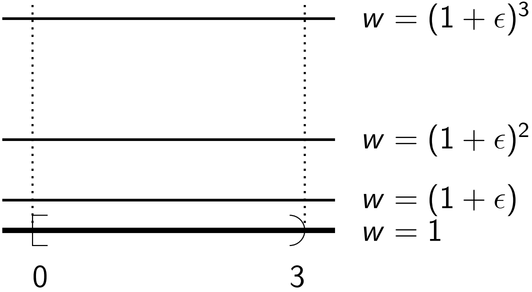
Weighted Unit-Length Intervals

- **Weighted Setting:** Intervals have weights, and we want to pick heavy intervals.
- **Key Idea:** Sub-divide the window into geometric weight classes.



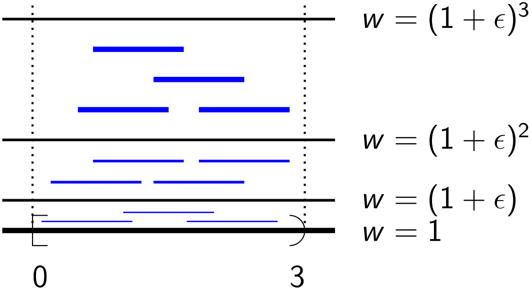
Weighted Unit-Length Intervals

- **Weighted Setting:** Intervals have weights, and we want to pick heavy intervals.
- **Key Idea:** Sub-divide the window into geometric weight classes.



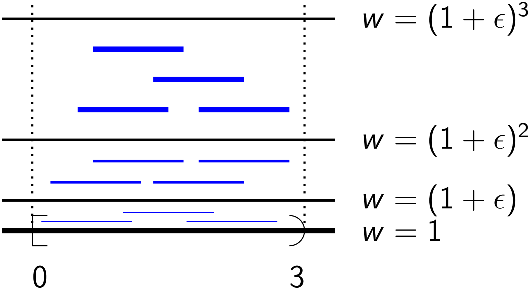
Weighted Unit-Length Intervals

- **Weighted Setting:** Intervals have weights, and we want to pick heavy intervals.
- **Key Idea:** Sub-divide the window into geometric weight classes.



Weighted Unit-Length Intervals

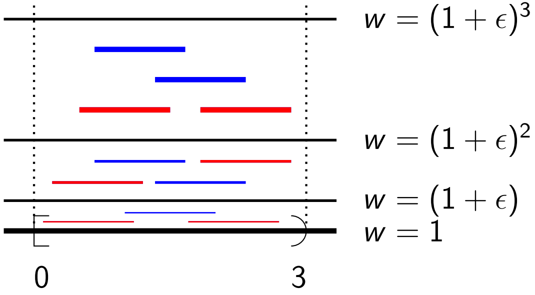
- **Weighted Setting:** Intervals have weights, and we want to pick heavy intervals.
- **Key Idea:** Sub-divide the window into geometric weight classes.



- Find the left and right most intervals in each weight class.

Weighted Unit-Length Intervals

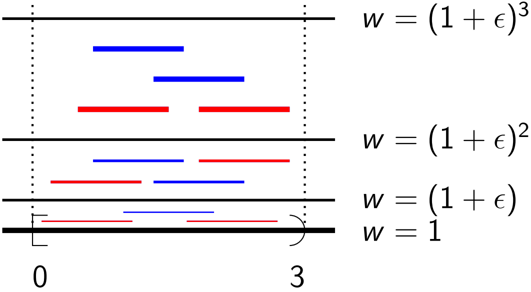
- **Weighted Setting:** Intervals have weights, and we want to pick heavy intervals.
- **Key Idea:** Sub-divide the window into geometric weight classes.



- Find the left and right most intervals in each weight class.

Weighted Unit-Length Intervals

- **Weighted Setting:** Intervals have weights, and we want to pick heavy intervals.
- **Key Idea:** Sub-divide the window into geometric weight classes.



- Find the left and right most intervals in each weight class.
- This gives a $(1+\epsilon)$-approximation within the heaviest $\log(1/\epsilon)/\epsilon$ weight classes.

State of the Art

		Insertion-Only		Insertion-Deletion	
		UB - $O_\epsilon(\lvert OPT \rvert)$	LB - $\Omega(n)$	UB - $\tilde{O}_\epsilon(\lvert OPT^* \rvert)$	LB - $\Omega(n)$
Unweighted	Unit-Length	$3/2$ [EHR12]	$3/2-\epsilon$ [EHR12]	$\mathbf{2}$	$2-\epsilon$ [BCW20]
	Variable-Length	2 [EHR12]	$2-\epsilon$ [EHR12]	$*$	$\boldsymbol{\Theta(1)}$
Weighted	Unit-Length	$\mathbf{3/2}+\epsilon$	$3/2-\epsilon$ [EHR12]	$\mathbf{2}+\epsilon$	$2-\epsilon$ [BCW20]
	Variable-Length	$*$	$\boldsymbol{\Theta(1)}$	$*$	$\boldsymbol{\Theta(1)}$

State of the Art

		Insertion-Only		Insertion-Deletion	
		UB - $O_\epsilon(\|OPT\|)$	LB - $\Omega(n)$	UB - $\tilde{O}_\epsilon(\|OPT^*\|)$	LB - $\Omega(n)$
Unweighted	Unit-Length	$3/2$ [EHR12]	$\mathbf{3/2} - \epsilon$ [EHR12]	$\mathbf{2}$	$2 - \epsilon$ [BCW20]
Unweighted	Variable-Length	2 [EHR12]	$2 - \epsilon$ [EHR12]	$*$	$\boldsymbol{\Theta(1)}$
Weighted	Unit-Length	$\mathbf{3/2} + \epsilon$	$3/2 - \epsilon$ [EHR12]	$\mathbf{2} + \epsilon$	$2 - \epsilon$ [BCW20]
Weighted	Variable-Length	$*$	$\boldsymbol{\Theta(1)}$	$*$	$\boldsymbol{\Theta(1)}$

Unweighted Unit-Length Intervals

Unweighted Unit-Length Intervals

- LB via communication complexity.

Unweighted Unit-Length Intervals

- LB via communication complexity.

Index Problem - INDEX(k)

Unweighted Unit-Length Intervals

- LB via communication complexity.

Index Problem - INDEX(k)

Alice

Unweighted Unit-Length Intervals

- LB via communication complexity.

Index Problem - INDEX(k)

Alice | Bob

Unweighted Unit-Length Intervals

- LB via communication complexity.

Index Problem - INDEX(k)

Alice	Bob	Out

Unweighted Unit-Length Intervals

- LB via communication complexity.

Index Problem - INDEX(k)

Alice	Bob	Out
X		

Unweighted Unit-Length Intervals

- LB via communication complexity.

Index Problem - INDEX(k)

Alice	Bob	Out
X	σ	

Unweighted Unit-Length Intervals

- LB via communication complexity.

Index Problem - INDEX(k)

Alice	Bob	Out
X	σ	Z

Unweighted Unit-Length Intervals

- LB via communication complexity.

Index Problem - INDEX(k)

Alice	$\xrightarrow{M}$	Bob	Out
X		σ	Z

Unweighted Unit-Length Intervals

- LB via communication complexity.

Index Problem - INDEX(k)

$$\frac{\text{Alice}}{X} \xrightarrow{M} \frac{\text{Bob}}{\sigma} \longrightarrow \frac{\text{Out}}{Z}$$

Unweighted Unit-Length Intervals

- LB via communication complexity.

Index Problem - INDEX(k)

$$\begin{tikzcd} \frac{\text{Alice}}{X} \arrow[r, bend left, "M"] & \frac{\text{Bob}}{\sigma} \arrow[r, bend left] & \frac{\text{Out}}{Z} \end{tikzcd}$$

- $X \in \{0,1\}^k$, $\sigma \in [k]$.

Unweighted Unit-Length Intervals

- LB via communication complexity.

Index Problem - INDEX(k)

$$\frac{\text{Alice}}{X} \xrightarrow{M} \frac{\text{Bob}}{\sigma} \longrightarrow \frac{\text{Out}}{Z}$$

- $X \in \{0,1\}^k$, $\sigma \in [k]$.
- $X[\sigma] = z \in \{0,1\}$.

Unweighted Unit-Length Intervals

- LB via communication complexity.

Index Problem - INDEX(k)

$$\begin{tikzcd} \dfrac{\text{Alice}}{X} \arrow[r, bend left, "M"] & \dfrac{\text{Bob}}{\sigma} \arrow[r, bend left] & \dfrac{\text{Out}}{Z} \end{tikzcd}$$

- $X \in \{0,1\}^k$, $\sigma \in [k]$.
- $X[\sigma] = z \in \{0,1\}$.
- **Goal:** Recover z using small messages.

Unweighted Unit-Length Intervals

- LB via communication complexity.

Index Problem - INDEX(k)

$$\begin{tikzcd} \dfrac{\text{Alice}}{X} \arrow[r, bend left, "M"] & \dfrac{\text{Bob}}{\sigma} \arrow[r, bend left] & \dfrac{\text{Out}}{Z} \end{tikzcd}$$

- $X \in \{0,1\}^k$, $\sigma \in [k]$.
- $X[\sigma] = z \in \{0,1\}$.
- **Goal:** Recover z using small messages.

Theorem

[KNR99] Any (randomised) protocol for INDEX(k) with success probability at least $2/3$ must send a message of size $\Omega(k)$.

Unweighted Unit-Length Intervals

Unweighted Unit-Length Intervals

- An algorithm for Unweighted Unit-Length Interval Selection can be used to solve INDEX(k).

Unweighted Unit-Length Intervals

- An algorithm for Unweighted Unit-Length Interval Selection can be used to solve INDEX(k).

- **Key Idea:** Encode $X \in \{0, 1\}^k$ into a clique gadget.

Unweighted Unit-Length Intervals

- An algorithm for Unweighted Unit-Length Interval Selection can be used to solve INDEX(k).

- **Key Idea:** Encode $X \in \{0, 1\}^k$ into a clique gadget.

Unweighted Unit-Length Intervals

- An algorithm for Unweighted Unit-Length Interval Selection can be used to solve INDEX(k).

- **Key Idea:** Encode $X \in \{0, 1\}^k$ into a clique gadget.

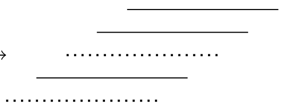
$$\begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Unweighted Unit-Length Intervals

- An algorithm for Unweighted Unit-Length Interval Selection can be used to solve INDEX(k).

- **Key Idea:** Encode $X \in \{0,1\}^k$ into a clique gadget.

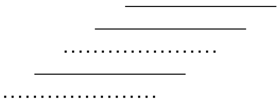
$$\begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \longrightarrow$$



Unweighted Unit-Length Intervals

- An algorithm for Unweighted Unit-Length Interval Selection can be used to solve INDEX(k).

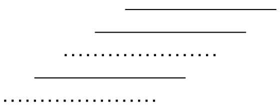
- **Key Idea:** Encode $X \in \{0, 1\}^k$ into a clique gadget.



Unweighted Unit-Length Intervals

- An algorithm for Unweighted Unit-Length Interval Selection can be used to solve INDEX(k).

- **Key Idea:** Encode $X \in \{0,1\}^k$ into a clique gadget.
- **Key Idea:** Use J to create two wing intervals.

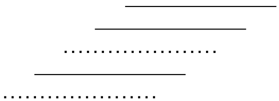


Unweighted Unit-Length Intervals

- An algorithm for Unweighted Unit-Length Interval Selection can be used to solve INDEX(k).

- **Key Idea:** Encode $X \in \{0,1\}^k$ into a clique gadget.
- **Key Idea:** Use J to create two wing intervals.

$J = 4$

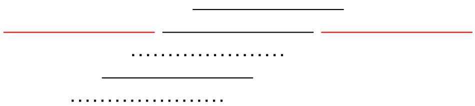


Unweighted Unit-Length Intervals

- An algorithm for Unweighted Unit-Length Interval Selection can be used to solve INDEX(k).

- **Key Idea:** Encode $X \in \{0,1\}^k$ into a clique gadget.
- **Key Idea:** Use J to create two wing intervals.

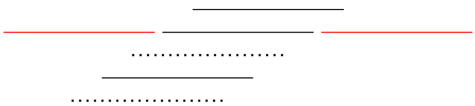
$J = 4 \longrightarrow$



Unweighted Unit-Length Intervals

- An algorithm for Unweighted Unit-Length Interval Selection can be used to solve INDEX(k).

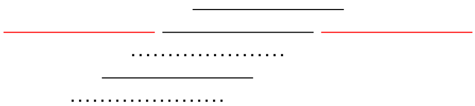
- **Key Idea:** Encode $X \in \{0,1\}^k$ into a clique gadget.
- **Key Idea:** Use J to create two wing intervals.



Unweighted Unit-Length Intervals

- An algorithm for Unweighted Unit-Length Interval Selection can be used to solve INDEX(k).

- **Key Idea:** Encode $X \in \{0,1\}^k$ into a clique gadget.
- **Key Idea:** Use J to create two wing intervals.
- This constructs a set of intervals which has an independent set of size 3 if $Z = 1$ and size 2 if $Z = 0$.



Unweighted Unit-Length Intervals

- An algorithm for Unweighted Unit-Length Interval Selection can be used to solve INDEX(k).

Unweighted Unit-Length Intervals

- An algorithm for Unweighted Unit-Length Interval Selection can be used to solve INDEX(k).

How does this apply to streaming?

Unweighted Unit-Length Intervals

- An algorithm for Unweighted Unit-Length Interval Selection can be used to solve INDEX(k).

How does this apply to streaming?

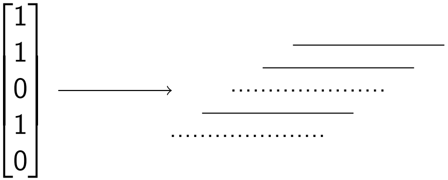
- **Alice:**

Unweighted Unit-Length Intervals

- An algorithm for Unweighted Unit-Length Interval Selection can be used to solve INDEX(k).

How does this apply to streaming?

- **Alice:**
  1. Use X to create the clique gadget.

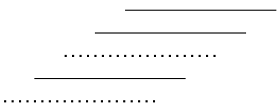


Unweighted Unit-Length Intervals

- An algorithm for Unweighted Unit-Length Interval Selection can be used to solve INDEX(k).

How does this apply to streaming?

- **Alice:**
  1. Use X to create the clique gadget.
  2. Run the streaming algorithm on this set of intervals and set the state of the algorithm to Bob.

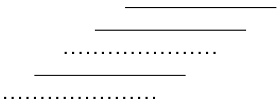


Unweighted Unit-Length Intervals

- An algorithm for Unweighted Unit-Length Interval Selection can be used to solve INDEX(k).

How does this apply to streaming?

- **Alice:**
  1. Use X to create the clique gadget.
  2. Run the streaming algorithm on this set of intervals and set the state of the algorithm to Bob.
- **Bob:**



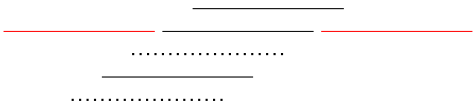
Unweighted Unit-Length Intervals

- An algorithm for Unweighted Unit-Length Interval Selection can be used to solve INDEX(k).

How does this apply to streaming?

- **Alice:**
  1. Use X to create the clique gadget.
  2. Run the streaming algorithm on this set of intervals and set the state of the algorithm to Bob.
- **Bob:**
  1. Use J to create the wing intervals.

$J = 4 \longrightarrow$

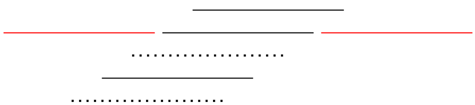


Unweighted Unit-Length Intervals

- An algorithm for Unweighted Unit-Length Interval Selection can be used to solve INDEX(k).

How does this apply to streaming?

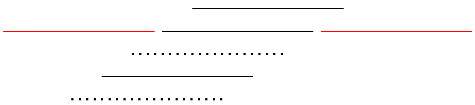
- **Alice:**
  1. Use X to create the clique gadget.
  2. Run the streaming algorithm on this set of intervals and set the state of the algorithm to Bob.
- **Bob:**
  1. Use J to create the wing intervals.
  2. Using the state sent by Alice, continue running the algorithm on the new intervals and output the result.



Unweighted Unit-Length Intervals

- An algorithm for Unweighted Unit-Length Interval Selection can be used to solve INDEX(k).

How does this apply to streaming?

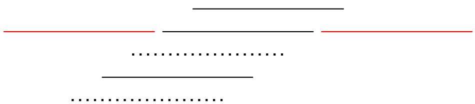


Unweighted Unit-Length Intervals

- An algorithm for Unweighted Unit-Length Interval Selection can be used to solve INDEX(k).

How does this apply to streaming?

- **Key Idea:** The message size is exactly the memory required for the algorithm.

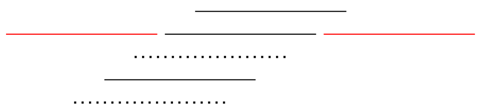


Unweighted Unit-Length Intervals

- An algorithm for Unweighted Unit-Length Interval Selection can be used to solve INDEX(k).

How does this apply to streaming?

- **Key Idea:** The message size is exactly the memory required for the algorithm.
- An algorithm to compute a better-than 3/2-approximation can be used to solve INDEX(k).

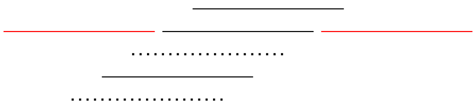


Unweighted Unit-Length Intervals

- An algorithm for Unweighted Unit-Length Interval Selection can be used to solve INDEX(k).

How does this apply to streaming?

- **Key Idea:** The message size is exactly the memory required for the algorithm.
- An algorithm to compute a better-than 3/2-approximation can be used to solve INDEX(k).
- By the INDEX(k) lower bound, such an algorithm must use space $\Omega(k) = \Omega(n)$.



State of the Art

		Insertion-Only		Insertion-Deletion	
		UB - $O_\epsilon(\lvert OPT\rvert)$	LB - $\Omega(n)$	UB - $\tilde{O}_\epsilon(\lvert OPT^*\rvert)$	LB - $\Omega(n)$
Unweighted	Unit-Length	$3/2$ [EHR12]	$\mathbf{3/2} - \epsilon$ [EHR12]	$\mathbf{2}$	$2 - \epsilon$ [BCW20]
	Variable-Length	2 [EHR12]	$2 - \epsilon$ [EHR12]	$*$	$\mathbf{\Theta(1)}$
Weighted	Unit-Length	$\mathbf{3/2} + \epsilon$	$3/2 - \epsilon$ [EHR12]	$\mathbf{2} + \epsilon$	$2 - \epsilon$ [BCW20]
	Variable-Length	$*$	$\mathbf{\Theta(1)}$	$*$	$\mathbf{\Theta(1)}$

State of the Art

		Insertion-Only		Insertion-Deletion	
		UB - $O_\epsilon(\lvert OPT\rvert)$	LB - $\Omega(n)$	UB - $\tilde{O}_\epsilon(\lvert OPT^*\rvert)$	LB - $\Omega(n)$
Unweighted	Unit-Length	$3/2$ [EHR12]	$3/2-\epsilon$ [EHR12]	$\mathbf{2}$	$2-\epsilon$ [BCW20]
	Variable-Length	2 [EHR12]	$2-\epsilon$ [EHR12]	$*$	$\boldsymbol{\Theta(1)}$
Weighted	Unit-Length	$\mathbf{3/2}+\epsilon$	$3/2-\epsilon$ [EHR12]	$\mathbf{2}+\epsilon$	$2-\epsilon$ [BCW20]
	Variable-Length	$*$	$\boldsymbol{\Theta(1)}$	$*$	$\boldsymbol{\Theta(1)}$

A Closing Thought

Communication Complexity can be used to achieve non-trivial lower bounds for streaming algorithms!

References I

Ainesh Bakshi, Nadiia Chepurko, and David P. Woodruff.
Weighted maximum independent set of geometric objects in turnstile streams.
In Jaroslaw Byrka and Raghu Meka, editors, Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference, volume 176 of LIPIcs, pages 64:1–64:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

Sergio Cabello and Pablo Pérez-Lantero.
Interval selection in the streaming model.
Theoretical Computer Science, 702:77–96, 2017.

References II

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

I. Kremer, N. Nisan, and D. Ron.
On randomized one-round communication complexity.
computational complexity, 8(1):21–49, 1999.