Interval Graphs in Data Streaming

Sergio Cabello

University of Ljubljana

Slovenia

Joint work with
Pablo Pérez Lantero, Universidad de Valparaíso (Chile)

Outline

- Interval selection problem
- Streaming model
- Our knowledge
- Deeper into some of the ideas
- Conclusions

Interval selection problem

Input: A set $\mathbb{I}$ of intervals in the real line
Task: find a largest cardinality $\mathbb{J} \subseteq \mathbb{I}$ of pairwise disjoint intervals.

- $\alpha(\mathbb{I})\ldots$ cardinality of optimal solution
- easy to find optimal solution in $O(|\mathbb{I}| \log |\mathbb{I}|)$

Streaming model

- Input intervals in an array $\mathbb{I}[1..\ell]$
- One pass over the array
- Little additional memory
- **Bounded universe**: endpoints in $\{1, \ldots, n\}$

Streaming model

- Input intervals in an array $\mathbb{I}[1..\ell]$
- One pass over the array
- Little additional memory
- **Bounded universe**: endpoints in $\{1, \ldots, n\}$

- What can we compute about $\alpha(\mathbb{I})$?
- What can we compute about max clique of $G(\mathbb{I})$?
- What can we compute about nb of edges of $G(\mathbb{I})$?

Exact solutions are impossible in $o(|\mathbb{I}|)$ memory
$\Rightarrow$ aim at **approximations**

Streaming model

- Input intervals in an array $\mathbb{I}[1..\ell]$
- One pass over the array
- Little additional memory
- **Bounded universe**: endpoints in $\{1, \ldots, n\}$

- What can we compute about $\alpha(\mathbb{I})$?
- What can we compute about max clique of $G(\mathbb{I})$?
- What can we compute about nb of edges of $G(\mathbb{I})$?

Exact solutions are impossible in $o(|\mathbb{I}|)$ memory
$\Rightarrow$ aim at **approximations**

Let's talk about $\alpha(\mathbb{I})$ ...

Our knowledge

- arbitrary intervals
  - 2-approximation using $O(\alpha(\mathbb{I}))$ space [EHR, CP]
  - 1.999-approximation impossible in $o(|\mathbb{I}|)$ space [EHR]
  - $(2+\varepsilon)$-estimate of $\alpha(\mathbb{I})$ using $O(\varepsilon^{-5}\log^6 n)$ space [CP]
  - 1.999-estimate of $\alpha(\mathbb{I})$ impossible using $o(n)$ space [CP]

[EHR] Emek, Halldórsson, and Rosén, ICALP 2012

[CP] Cabello and Pérez Lantero, WADS 2015

Our knowledge

- arbitrary intervals
  - 2-approximation using $O(\alpha(\mathbb{I}))$ space [EHR, CP]
  - 1.999-approximation impossible in $o(|\mathbb{I}|)$ space [EHR]
  - $(2+\varepsilon)$-estimate of $\alpha(\mathbb{I})$ using $O(\varepsilon^{-5}\log^6 n)$ space [CP]
  - 1.999-estimate of $\alpha(\mathbb{I})$ impossible using $o(n)$ space [CP]
- same-size intervals
  - $(3/2)$-approximation with $O(\alpha(\mathbb{I}))$ space for **proper** intervals [EHR]
  - 1.499-approximation impossible in $o(|\mathbb{I}|)$ space [EHR]
  - simpler $(3/2)$-approximation with $O(\alpha(\mathbb{I}))$ space [CP]
  - $(3/2+\varepsilon)$-estimate of $\alpha(\mathbb{I})$ using $O(\varepsilon^{-2}\log(1/\varepsilon)+\log n)$ space [CP]
  - 1.499-estimate of $\alpha(\mathbb{I})$ impossible using $o(n)$ space [CP]

[EHR] Emek, Halldórsson, and Rosén, ICALP 2012

[CP] Cabello and Pérez Lantero, WADS 2015

My points

Basic problems are not simple in the streaming model

Intersection graphs in data stream model

I found the techniques in data streaming cool

Outline

- Interval selection problem
- Streaming model
- Our knowledge
- **Deeper into some of the ideas** – Same size intervals
- Conclusions

Technical content of my talk

- only **same-size** intervals
- all intervals closed of length λ
- endpoints in $[n] = \{1, 2, \ldots n\}$

Technical content of my talk

- only **same-size** intervals
- all intervals closed of length λ
- endpoints in $[n] = \{1, 2, \ldots n\}$

Theorem

$(3/2)$-approximation using $O(\alpha(\mathbb{I}))$ space.
Deterministic. Even comparison model.

Theorem

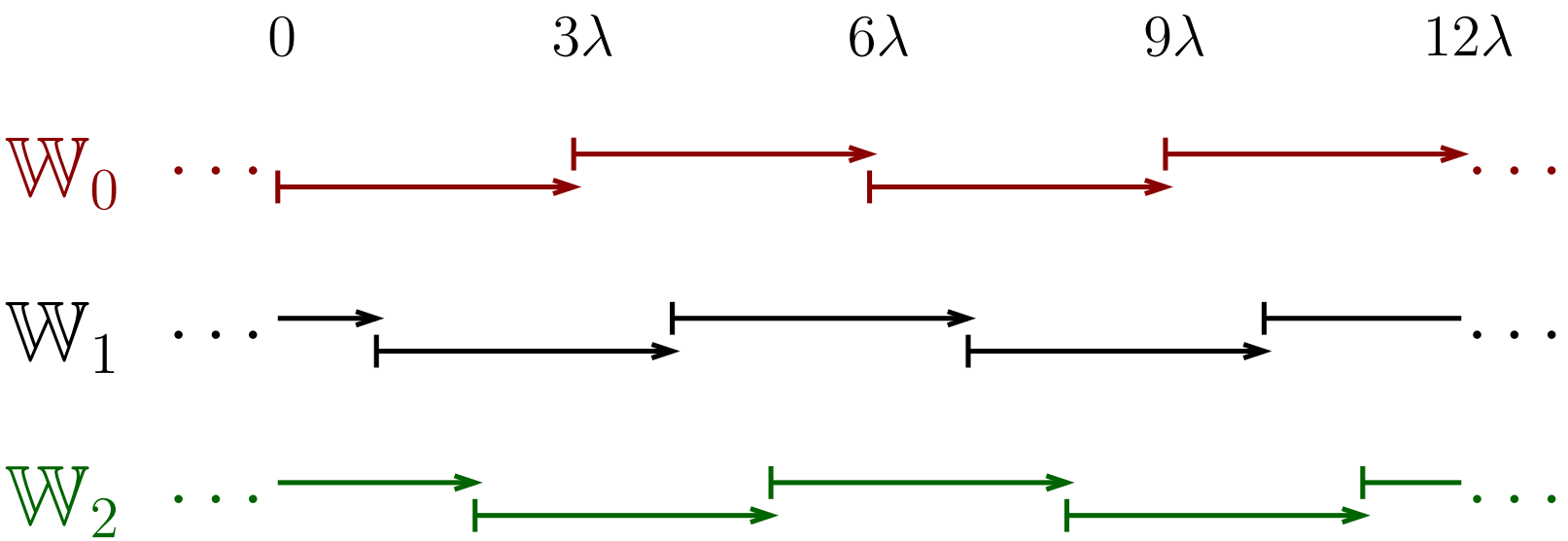
Randomized. Using $O(\varepsilon^{-2} \log(1/\varepsilon) + \log n)$ space we get an estimate $\hat{\alpha}$ to $\alpha = \alpha(\mathbb{I})$ such that

$$\Pr\left[\alpha \leq \hat{\alpha} \leq \left(\frac{3}{2} + \varepsilon\right)\alpha\right] \geq \frac{3}{4}.$$

Shifting technique

- Make 3 partitions of the line into windows of length 3λ

$$\mathbb{W}_0 = \{\ldots, [0, 3\lambda), [3\lambda, 6\lambda), [6\lambda, 9\lambda), \ldots\}$$
$$\mathbb{W}_1 = \{\ldots, [\lambda, 4\lambda), [4\lambda, 7\lambda), [7\lambda, 10\lambda), \ldots\}$$
$$\mathbb{W}_2 = \{\ldots, [2\lambda, 5\lambda), [5\lambda, 8\lambda), [8\lambda, 11\lambda), \ldots\}$$



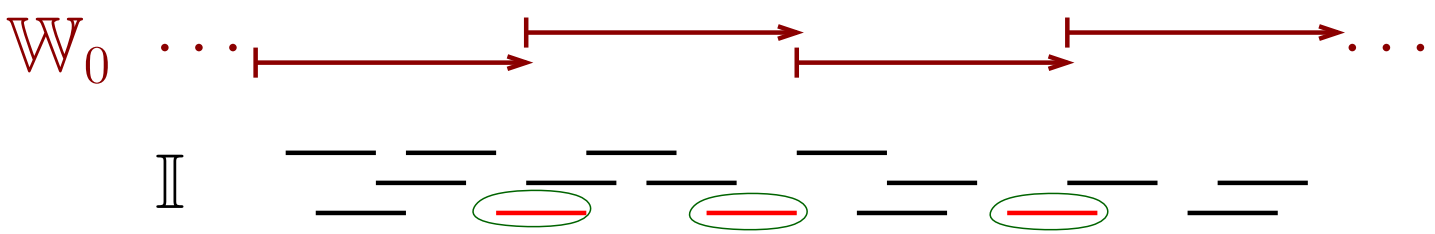
Shifting technique

- for $a = 0, 1, 2$ we have the partition

$$\mathbb{W}_a = \{\ldots, [a, (3+a)\lambda), [(3+a)\lambda, (6+a)\lambda), \ldots\}$$

and the subproblems

$$\mathbb{I}_a = \{I \in \mathbb{I} \mid I \text{ contained in some window of } \mathbb{W}_a\}$$



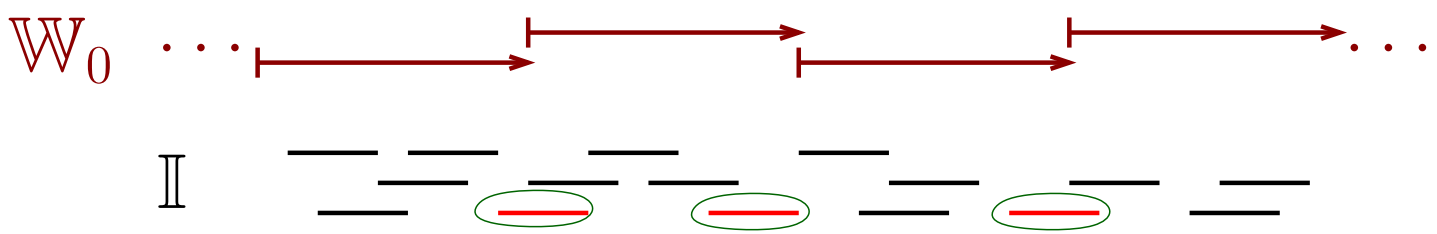
Shifting technique

- for $a = 0, 1, 2$ we have the partition

$$\mathbb{W}_a = \{\ldots, [a, (3+a)\lambda), [(3+a)\lambda, (6+a)\lambda), \ldots\}$$

and the subproblems

$$\mathbb{I}_a \;=\; \{I \in \mathbb{I} \mid I \text{ contained in some window of } \mathbb{W}_a\}$$



Theorem (Hochbaum, Maass; JACM'85)

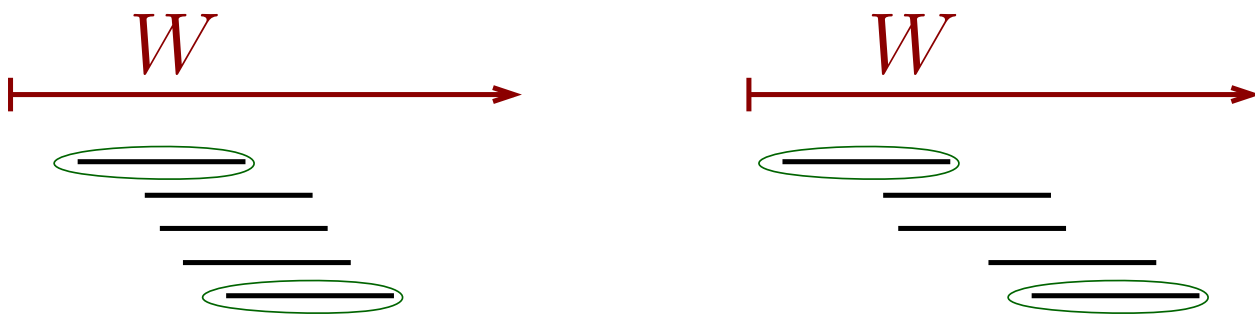
$$\max\{\alpha(\mathbb{I}_0), \alpha(\mathbb{I}_1), \alpha(\mathbb{I}_2)\} \;\geq\; \frac{2}{3}\alpha(\mathbb{I})$$

Approximation algorithm – Same size

- for $a = 0, 1, 2$ we want a solution of size $\alpha(\mathbb{I}_a)$
- run each $a = 0, 1, 2$ in parallel
- fix $a = 0$, for simplicity (so $\mathbb{W}_0$ and $\mathbb{I}_0$)

Approximation algorithm – Same size

- for $a = 0, 1, 2$ we want a solution of size $\alpha(\mathbb{I}_a)$
- run each $a = 0, 1, 2$ in parallel
- fix $a = 0$, for simplicity (so $\mathbb{W}_0$ and $\mathbb{I}_0$)
- each window $W \in \mathbb{W}_0$
  - semiopen of length 3λ
  - contains 0, 1 or 2 disjoint input intervals



- N_t = number of windows in $\mathbb{W}_0$ that contain $\geq t$ disjoint input intervals.

$$\alpha(\mathbb{I}_0) \;=\; N_1 + N_2$$

Approximation algorithm – Same size

- keep non-empty windows of $\mathbb{W}_0$ in a BST and counters N_1, N_2
- for each non-empty window W of $\mathbb{W}_0$
  - store leftmost and rightmost interval contained in W
  - if they are disjoint, W counts towards N_2

Approximation algorithm – Same size

- keep non-empty windows of $\mathbb{W}_0$ in a BST and counters N_1, N_2
- for each non-empty window W of $\mathbb{W}_0$
  - store leftmost and rightmost interval contained in W
  - if they are disjoint, W counts towards N_2
- we use $O(\alpha(\mathbb{I}_0)) = O(\alpha(\mathbb{I}))$ space
- for each new interval $I \in \mathbb{I}$
  - if $I \in \mathbb{I}_0$
    - compute $W \in \mathbb{W}_0$ that contains I
    - search W in the BST
    - if W not present in BST, insert it and increase N_1
    - if W present, check whether we have to increase N_2
    - update lefmost and rightmost interval of W

Approximation algorithm – Same size

Theorem

(3/2)-*approximation using* $O(\alpha(\mathbb{I}))$ *space.*
Deterministic. Even comparison model.

No real use of data streaming techniques...
If $\alpha(\mathbb{I}) = \Theta(|\mathbb{I}|)$, a lot of space.

Estimate α – Same size

- independent estimations of $\alpha(\mathbb{I}_0), \alpha(\mathbb{I}_1), \alpha(\mathbb{I}_2)$
- run each $a = 0, 1, 2$ in parallel
- fix $a = 0$, for simplicity (so $\mathbb{W}_0$ and $\mathbb{I}_0$)
- N_t = number of windows in $\mathbb{W}_0$ that contain $\geq t$ disjoint input intervals.

$$\alpha(\mathbb{I}_0) \;=\; N_1 + N_2$$

- Aim: estimate N_1 and N_2
- we need a couple of tools

Tool 1: Distinct elements

- Input: stream of elements $a_1, a_2, \ldots, a_\ell$ from $\{1, \ldots, u\}$
- Task: estimate number of distinct elements in the stream

$$|\{a_1, \ldots, a_\ell\}|$$

- A classical problem in data streaming
- Probabilistic $(1+\varepsilon)$-estimate using $O(\varepsilon^{-2} + \log u)$ space by Kane, Nelson, Woodruff, PODS 2010

Tool 1: Distinct elements

- Input: stream of elements $a_1, a_2, \ldots, a_\ell$ from $\{1, \ldots, u\}$
- Task: estimate number of distinct elements in the stream

$$|\{a_1, \ldots, a_\ell\}|$$

- A classical problem in data streaming
- Probabilistic $(1+\varepsilon)$-estimate using $O(\varepsilon^{-2} + \log u)$ space by Kane, Nelson, Woodruff, PODS 2010

- We can $(1+\varepsilon)$-**estimate** N_1 using distinct elements as black box
- The stream $I_1, I_2, \ldots$ defines a stream $W_1, W_2, \ldots$ of windows from $\mathbb{W}_0$
- count number of distinct elements in $W_1, W_2, \ldots$

Tool 2: Near-uniform sampling

- Input: stream of elements $a_1, a_2, \ldots, a_\ell$ from $\{1, \ldots, u\}$
- Task: sample one element of the stream, each with equal probability
- Withough multiplicity; each element sampled with probability

$$1/|\{a_1, \ldots, a_\ell\}|$$

- The sampled element should be chosen the first time it is seen

Tool 2: Near-uniform sampling

- Input: stream of elements $a_1, a_2, \ldots, a_\ell$ from $\{1, \ldots, u\}$
- Task: sample one element of the stream, each with equal probability
- Withough multiplicity; each element sampled with probability

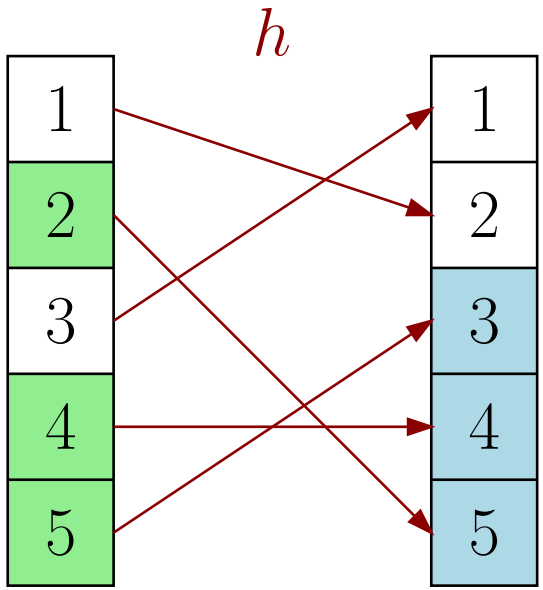
$$1/|\{a_1, \ldots, a_\ell\}|$$

- The sampled element should be chosen the first time it is seen

- Approach: choose random permutation $h : \{1, \ldots, u\} \to \{1, \ldots, u\}$
- Choose the element minimizing

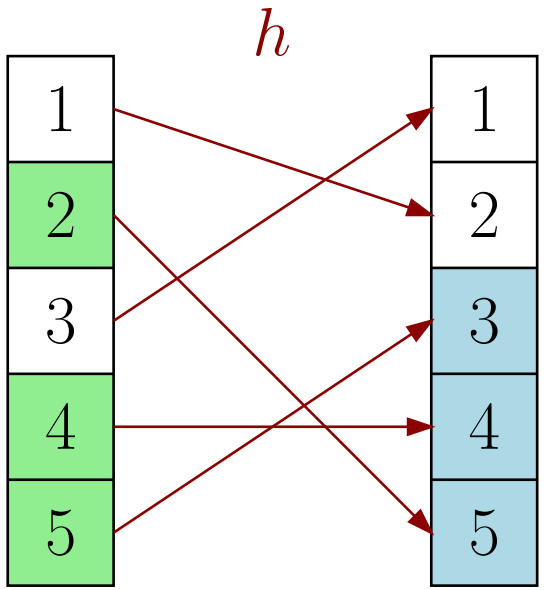
$$h(a_1), h(a_2), h(a_3), \ldots$$

Tool 2: Near-uniform sampling



$4, 4, 4, 2, 4, 5, 2, 2,$
$2, 4, 5, 5, 2, \ldots$

Tool 2: Near-uniform sampling



$4, 4, 4, 2, 4, 5, 2, 2,$
$2, 4, 5, 5, 2, \ldots$

- Problem: encoding a random h is expensive
- Construct h using hash functions & sacrifice uniformity

Tool 2: Near-uniform sampling

Lemma

There exists a family of permutations $\mathcal{H}(u, \varepsilon) = \{h : [u] \to [u]\}$ *such that:*

- $\mathcal{H}(u, \varepsilon)$ *has* $n^{O(\log(1/\varepsilon))}$ *permutations;*
- *a random element of* $\mathcal{H}(u, \varepsilon)$ *can be chosen in* $O(\log(1/\varepsilon))$ *time;*
- *for* $h \in \mathcal{H}(u, \varepsilon)$ *and* $a, b \in [u]$*, we can decide with* $O(\log(1/\varepsilon))$ *arithmetic operations whether* $h(a) < h(b)$
- $$\forall A \subseteq [u], a \in A : \quad \frac{1-\varepsilon}{|A|} \leq \Pr_{h \in \mathcal{H}}[h(a) = \min h(A)] \leq \frac{1+\varepsilon}{|A|}.$$

Tool 2: Near-uniform sampling

Lemma

There exists a family of permutations $\mathcal{H}(u, \varepsilon) = \{h : [u] \to [u]\}$ *such that:*

- $\mathcal{H}(u, \varepsilon)$ *has* $n^{O(\log(1/\varepsilon))}$ *permutations;*
- *a random element of* $\mathcal{H}(u, \varepsilon)$ *can be chosen in* $O(\log(1/\varepsilon))$ *time;*
- *for* $h \in \mathcal{H}(u, \varepsilon)$ *and* $a, b \in [u]$*, we can decide with* $O(\log(1/\varepsilon))$ *arithmetic operations whether* $h(a) < h(b)$
- $$\forall A \subseteq [u], a \in A : \quad \frac{1-\varepsilon}{|A|} \leq \Pr_{h \in \mathcal{H}}[h(a) = \min h(A)] \leq \frac{1+\varepsilon}{|A|}.$$

Useful for near-uniform sampling
Keyword: ε-min-wise independent hash functions
Based on Indyk'01, used by Broder, Charikar, Datar, Frieze, Mitzenmacher, Muthukrishnan.

Estimate N_2

- Take $\Theta(\varepsilon^{-2})$ samples of the windows of $\mathbb{W}_0$ that contain some interval
- Check the proportion of them that contain 2 disjoint input intervals
- Good estimate for the ratio N_2/N_1
- We already have a $(1+\varepsilon)$-estimate of N_1
- We estimate N_2 with an error of εN_1.
- We get a $(1+\varepsilon)$-estimate for $N_1 + N_2 = \alpha(\mathbb{I}_0)$

Theorem

Randomized. Using $O(\varepsilon^{-2}\log(1/\varepsilon) + \log n)$ *space we get an estimate* $\hat{\alpha}$ *to* $\alpha = \alpha(\mathbb{I})$ *such that*

$$\Pr\left[\alpha \leq \hat{\alpha} \leq \left(\frac{3}{2}+\varepsilon\right)\alpha\right] \geq \frac{3}{4}.$$

Conclusions

- Interval selection easy
- Many variations are not easy:
  - Streaming
  - Stochastic
  - Dynamic
- Other problems in streaming for intervals
  - Nb of pairs of intervals intersecting
    - ⋆ unclear if solved
  - Clique
    - ⋆ doable via frequency estimation and dyadic trees

The end

Thanks :)