Space Optimal Vertex Cover in Dynamic Streams
(with an overview of graph streaming)

Kheeran K. Naidu

University of Bristol

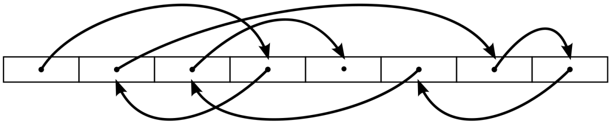
kheeran.naidu@bristol.ac.uk

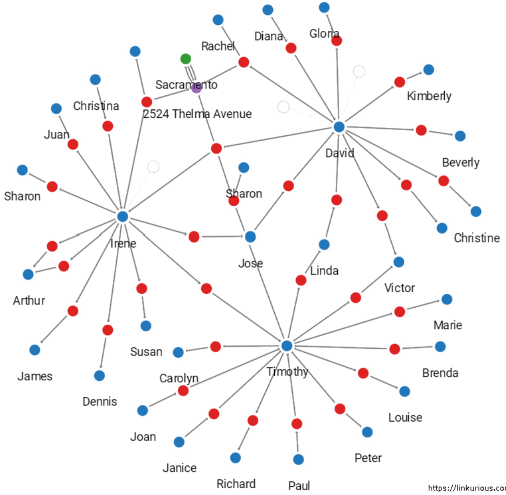
Joint work with Vihan Shah (Rutgers University)

Classic Setting

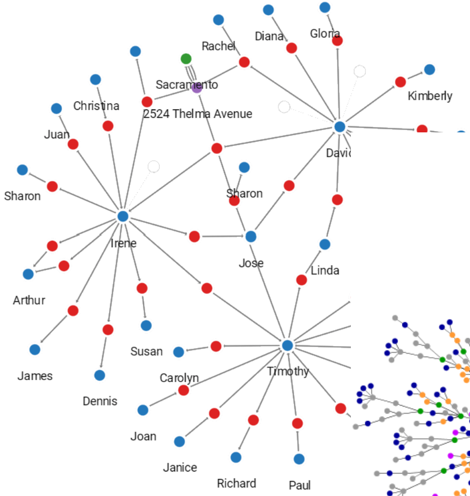
Assumption

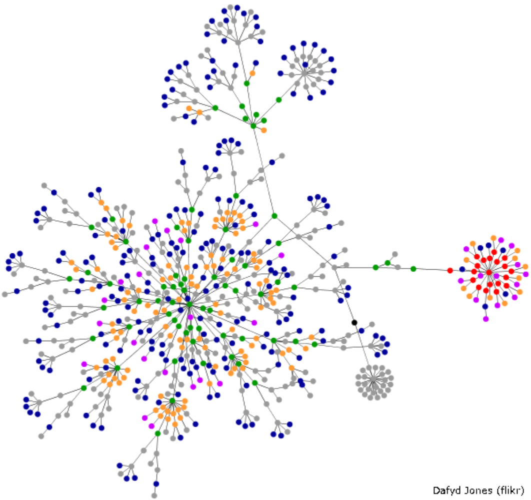
Classical algorithms rely on the assumption that they have a random access to the input of the algorithm



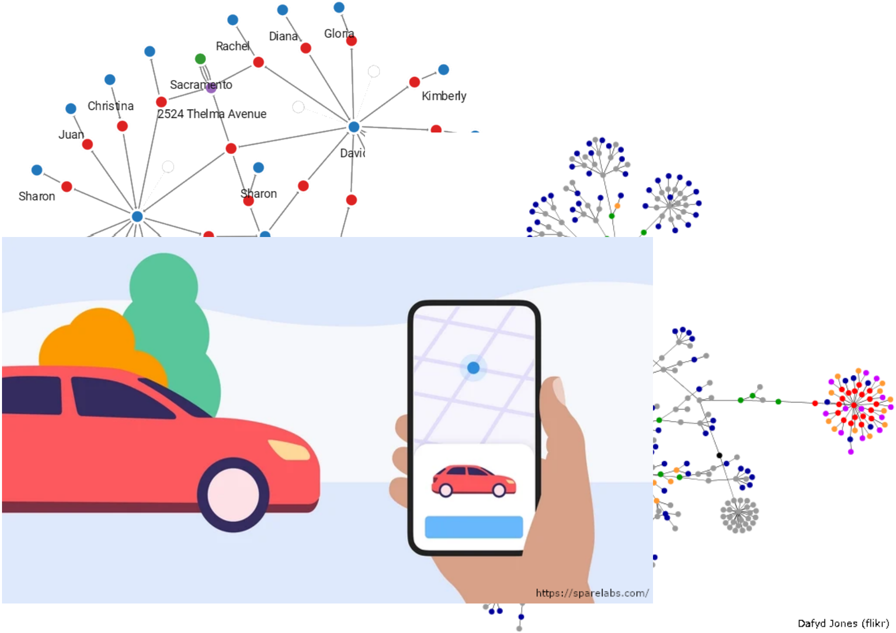


https://linkurious.com/





Dafyd Jones (flickr)

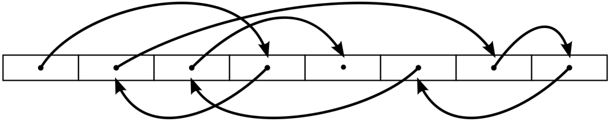


Dafyd Jones (flickr)

Dafyd Jones (flikr)

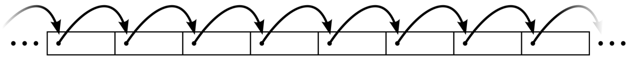
Assumption (Infeasible)

Classical algorithms rely on the assumption that they have a random access to the input of the algorithm



Definition (Graph Streaming)

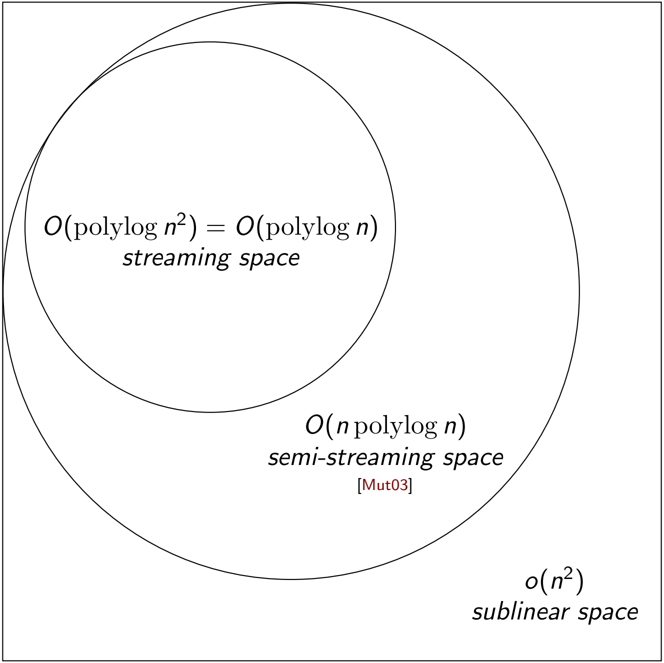
A n-vertex graph is presented as a sequence of edges to an algorithm uses space **sublinear** in the size of the input ($o(n^2)$).



$o(n^2)$

sublinear space


$O(\text{polylog}\, n^2) = O(\text{polylog}\, n)$
streaming space
$o(n^2)$
sublinear space

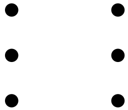

$O(\text{polylog}\, n^2) = O(\text{polylog}\, n)$
streaming space
$O(n\, \text{polylog}\, n)$
semi-streaming space
[Mut03]
$o(n^2)$
sublinear space

Insertion-Only [FKM+04]

Sliding Window [CMS13]

Dynamic [AGM12]

Insertion-Only [FKM$^+$04]

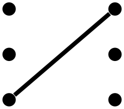


Sliding Window [CMS13]

Dynamic [AGM12]

Insertion-Only [FKM+04]

$\boxed{e_1}$

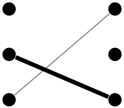


Sliding Window [CMS13]

Dynamic [AGM12]

Insertion-Only [FKM+04]

e_1	e_2

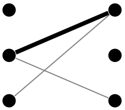


Sliding Window [CMS13]

Dynamic [AGM12]

Insertion-Only [FKM+04]

e_1	e_2	e_3

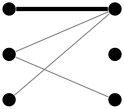


Sliding Window [CMS13]

Dynamic [AGM12]

Insertion-Only [FKM+04]

e_1	e_2	e_3	e_4

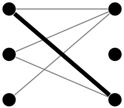


Sliding Window [CMS13]

Dynamic [AGM12]

Insertion-Only [FKM+04]

e_1	e_2	e_3	e_4	e_5

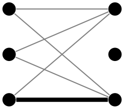


Sliding Window [CMS13]

Dynamic [AGM12]

Insertion-Only [FKM+04]

e_1	e_2	e_3	e_4	e_5	e_6

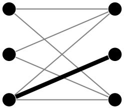


Sliding Window [CMS13]

Dynamic [AGM12]

Insertion-Only [FKM+04]

e_1	e_2	e_3	e_4	e_5	e_6	e_7

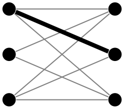


Sliding Window [CMS13]

Dynamic [AGM12]

Insertion-Only [FKM+04]

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8

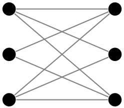


Sliding Window [CMS13]

Dynamic [AGM12]

Insertion-Only [FKM+04] (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8

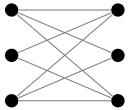


Sliding Window [CMS13]

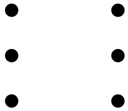
Dynamic [AGM12]

Insertion-Only [FKM+04] (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



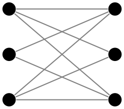
Sliding Window [CMS13]



Dynamic [AGM12]

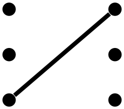
Insertion-Only [FKM+04] (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



Sliding Window [CMS13]

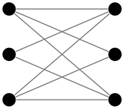
e_1



Dynamic [AGM12]

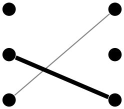
Insertion-Only [FKM+04] (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



Sliding Window [CMS13]

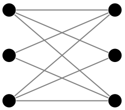
e_1	e_2



Dynamic [AGM12]

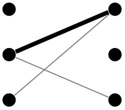
Insertion-Only [FKM+04] (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



Sliding Window [CMS13]

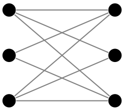
e_1	e_2	e_3



Dynamic [AGM12]

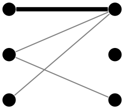
Insertion-Only [FKM+04] (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



Sliding Window [CMS13]

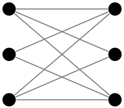
e_1	e_2	e_3	e_4



Dynamic [AGM12]

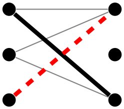
Insertion-Only [FKM+04] (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



Sliding Window [CMS13]

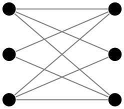
e_1	e_2	e_3	e_4	e_5



Dynamic [AGM12]

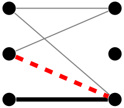
Insertion-Only [FKM+04] (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



Sliding Window [CMS13]

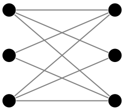
e_1	e_2	e_3	e_4	e_5	e_6



Dynamic [AGM12]

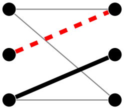
Insertion-Only [FKM+04] (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



Sliding Window [CMS13]

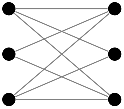
e_1	e_2	e_3	e_4	e_5	e_6	e_7



Dynamic [AGM12]

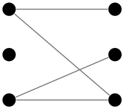
Insertion-Only [FKM+04] (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



Sliding Window [CMS13] (infinite stream)

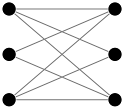
e_1	e_2	e_3	e_4	e_5	e_6	e_7	$\cdots$



Dynamic [AGM12]

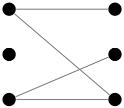
Insertion-Only [FKM+04] (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8

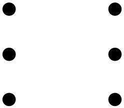


Sliding Window [CMS13] (infinite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	$\cdots$

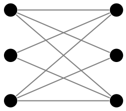


Dynamic [AGM12]



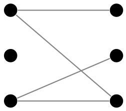
Insertion-Only [FKM+04] (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



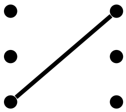
Sliding Window [CMS13] (infinite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	$\cdots$



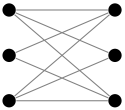
Dynamic [AGM12]

e_1



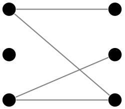
Insertion-Only [FKM+04] (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



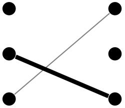
Sliding Window [CMS13] (infinite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	$\cdots$



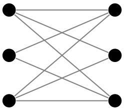
Dynamic [AGM12]

e_1	e_2



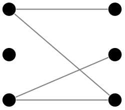
Insertion-Only [FKM+04] (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



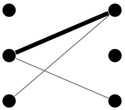
Sliding Window [CMS13] (infinite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	$\cdots$



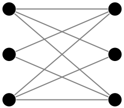
Dynamic [AGM12]

e_1	e_2	e_3



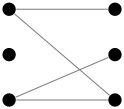
Insertion-Only [FKM+04] (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



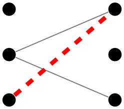
Sliding Window [CMS13] (infinite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	$\cdots$



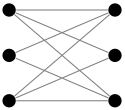
Dynamic [AGM12]

e_1	e_2	e_3	$\overline{e_1}$



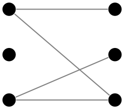
Insertion-Only [FKM+04] (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



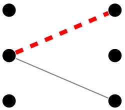
Sliding Window [CMS13] (infinite stream)

e_1	e_2	e_3	$\underline{e_4}$	$\underline{e_5}$	$\underline{e_6}$	$\underline{e_7}$	$\cdots$



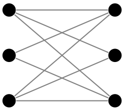
Dynamic [AGM12]

e_1	e_2	e_3	$\overline{e_1}$	$\overline{e_3}$



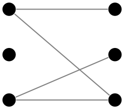
Insertion-Only [FKM+04] (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



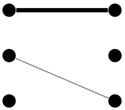
Sliding Window [CMS13] (infinite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	$\cdots$



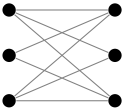
Dynamic [AGM12]

e_1	e_2	e_3	$\overline{e_1}$	$\overline{e_3}$	e_4



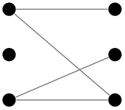
Insertion-Only [FKM+04] (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



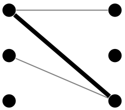
Sliding Window [CMS13] (infinite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	$\cdots$



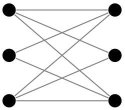
Dynamic [AGM12]

e_1	e_2	e_3	$\overline{e_1}$	$\overline{e_3}$	e_4	e_5



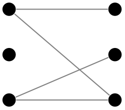
Insertion-Only [FKM+04] (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



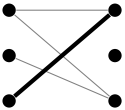
Sliding Window [CMS13] (infinite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	$\cdots$



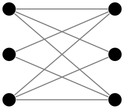
Dynamic [AGM12]

e_1	e_2	e_3	$\overline{e_1}$	$\overline{e_3}$	e_4	e_5	e_1



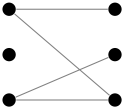
Insertion-Only [FKM+04] (finite stream)

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



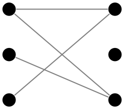
Sliding Window [CMS13] (infinite stream)

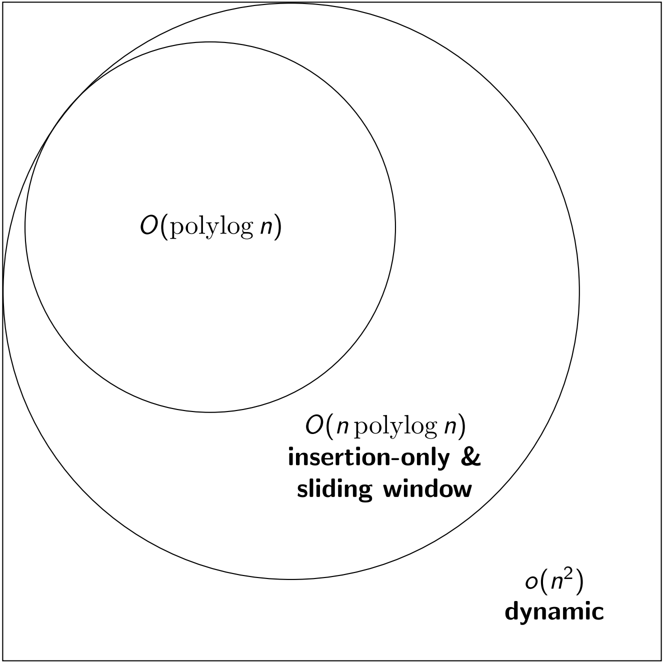
e_1	e_2	e_3	e_4	e_5	e_6	e_7	$\cdots$



Dynamic [AGM12] (finite stream)

e_1	e_2	e_3	$\overline{e_1}$	$\overline{e_3}$	e_4	e_5	e_1




$O(\mathrm{polylog}\, n)$
$O(n\, \mathrm{polylog}\, n)$
insertion-only &
sliding window
$o(n^2)$
dynamic

Simple and Powerful

Simple and Powerful

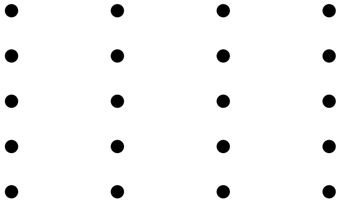
GREEDY Matching:

1. Add edge if neither endpoint is matched

Simple and Powerful

GREEDY Matching:

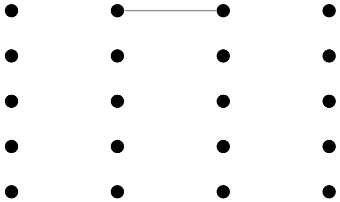
1. Add edge if neither endpoint is matched



Simple and Powerful

GREEDY Matching:

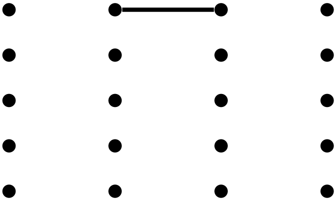
1. Add edge if neither endpoint is matched



Simple and Powerful

GREEDY Matching:

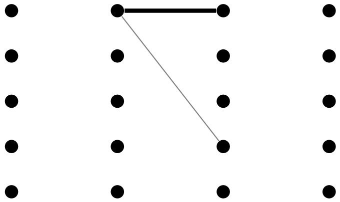
1. Add edge if neither endpoint is matched



Simple and Powerful

GREEDY Matching:

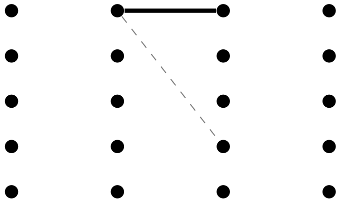
1. Add edge if neither endpoint is matched



Simple and Powerful

GREEDY Matching:

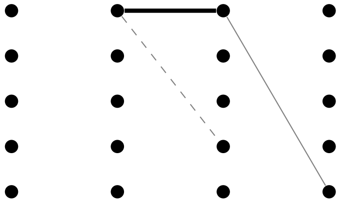
1. Add edge if neither endpoint is matched



Simple and Powerful

GREEDY Matching:

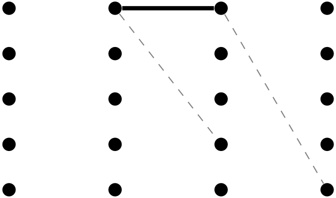
1. Add edge if neither endpoint is matched



Simple and Powerful

GREEDY Matching:

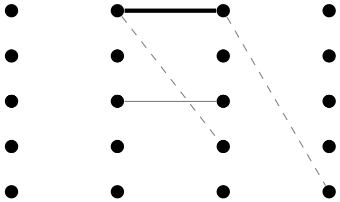
1. Add edge if neither endpoint is matched



Simple and Powerful

GREEDY Matching:

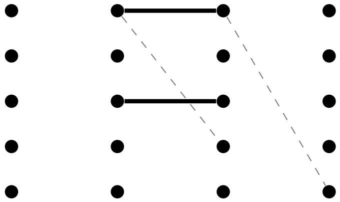
1. Add edge if neither endpoint is matched



Simple and Powerful

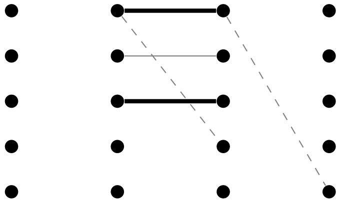
GREEDY Matching:

1. Add edge if neither endpoint is matched



GREEDY Matching:

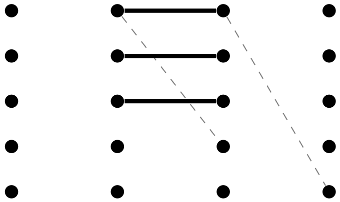
1. Add edge if neither endpoint is matched



Simple and Powerful

GREEDY Matching:

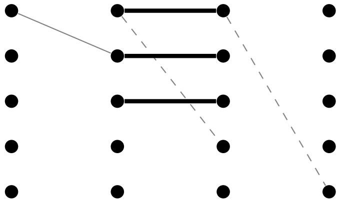
1. Add edge if neither endpoint is matched



Simple and Powerful

GREEDY Matching:

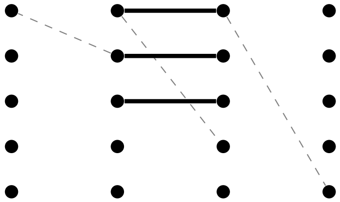
1. Add edge if neither endpoint is matched



Simple and Powerful

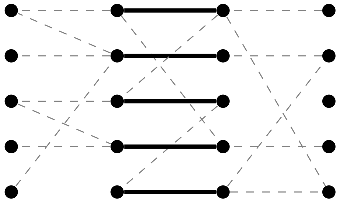
GREEDY Matching:

1. Add edge if neither endpoint is matched



GREEDY Matching:

1. Add edge if neither endpoint is matched

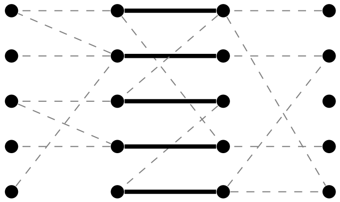


Simple and Powerful

GREEDY Matching:

1. Add edge if neither endpoint is matched

- Maximal

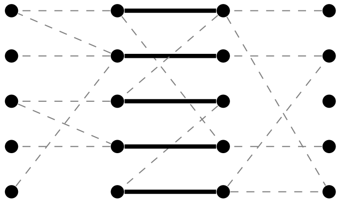


Simple and Powerful

GREEDY Matching:

1. Add edge if neither endpoint is matched

- Maximal
- 2-approximation

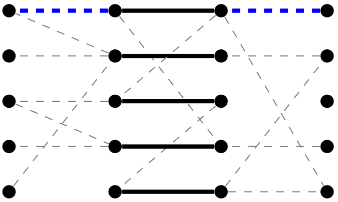


Simple and Powerful

GREEDY Matching:

1. Add edge if neither endpoint is matched

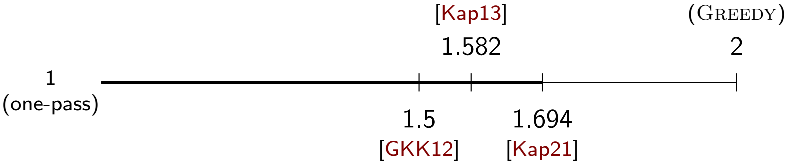
- Maximal
- 2-approximation



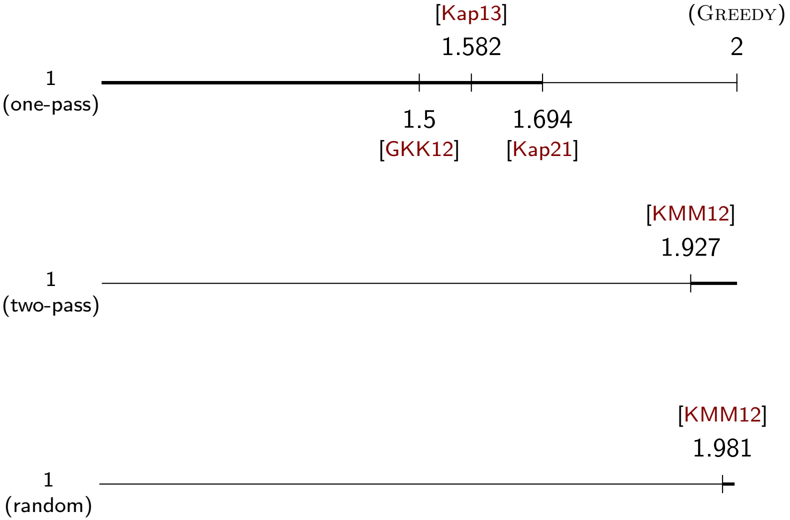
Insertion-Only

Insertion-Only

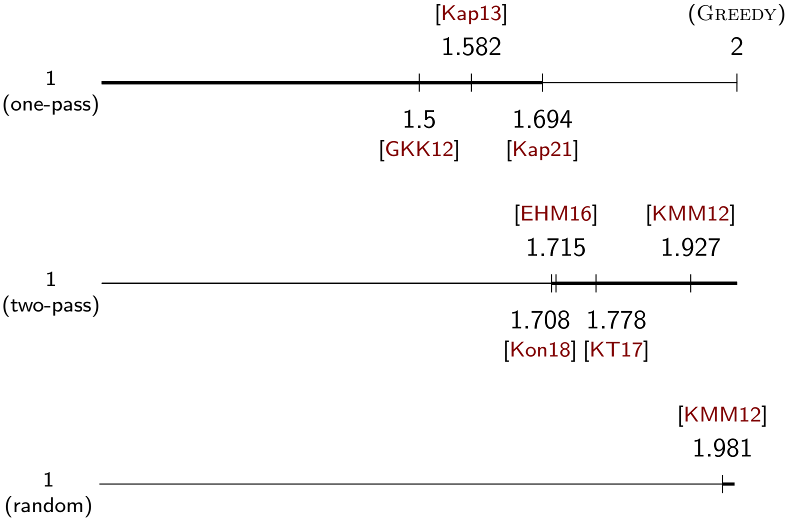
Insertion-Only



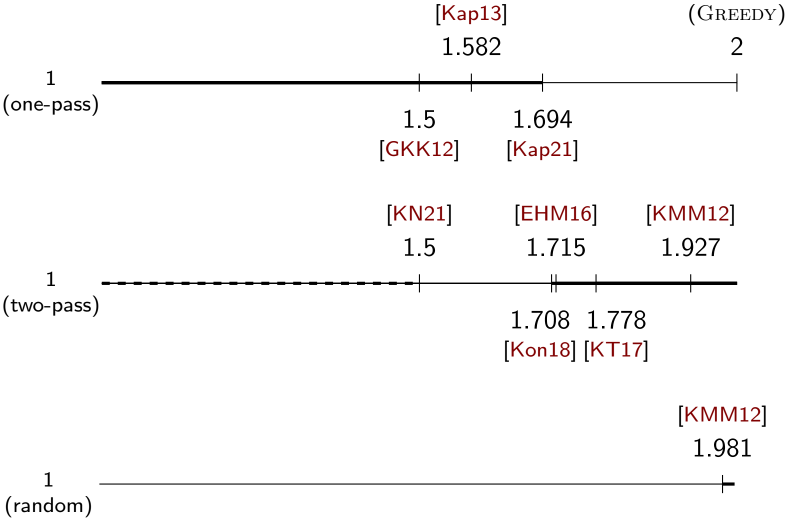
Insertion-Only



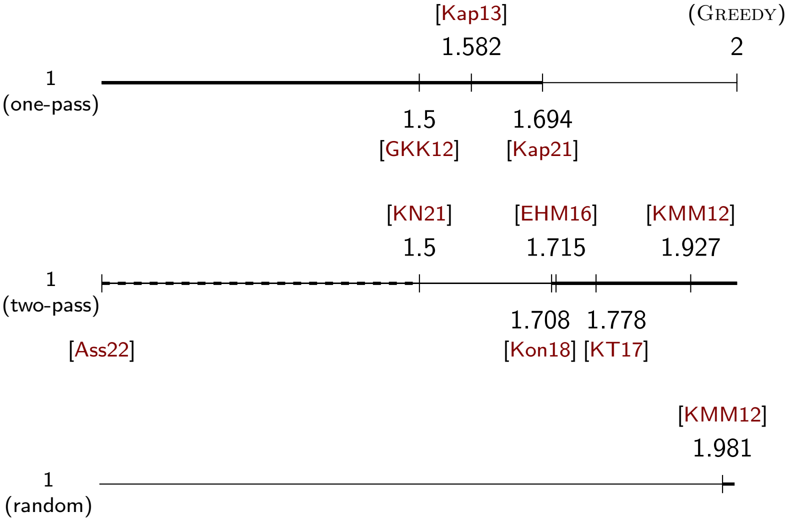
Insertion-Only



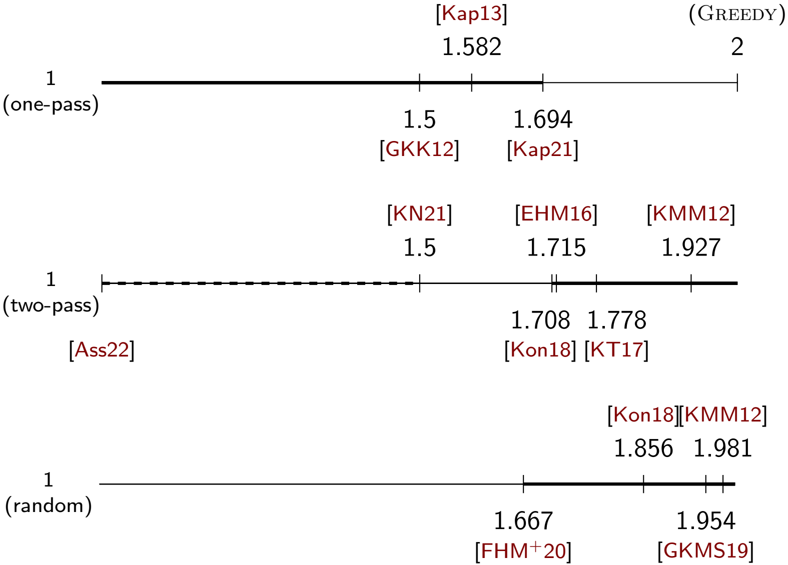
Insertion-Only



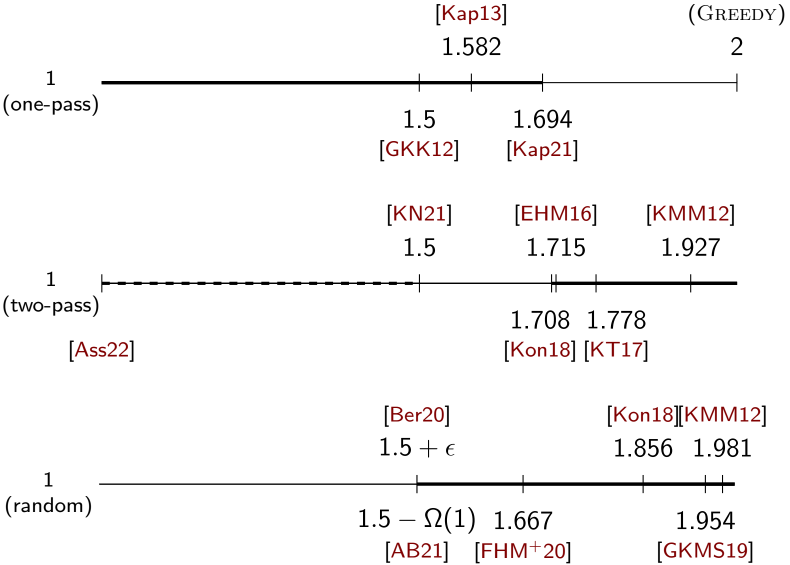
Insertion-Only



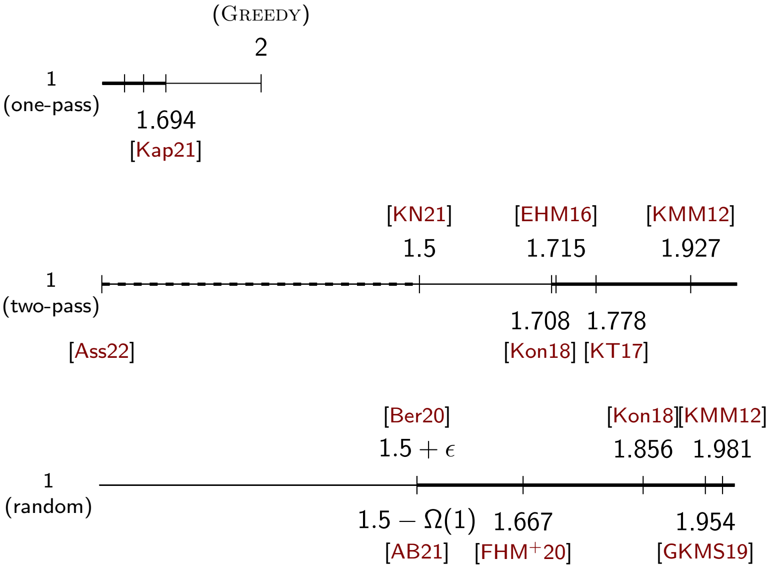
Insertion-Only



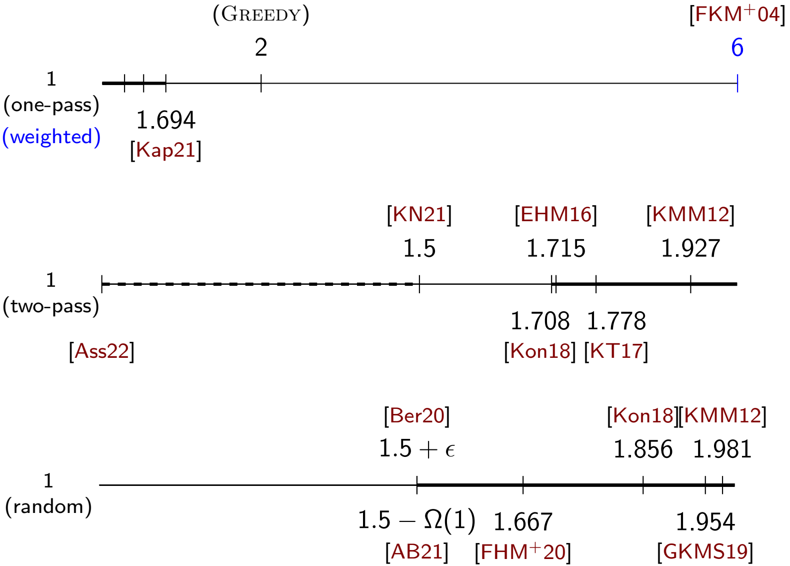
Insertion-Only



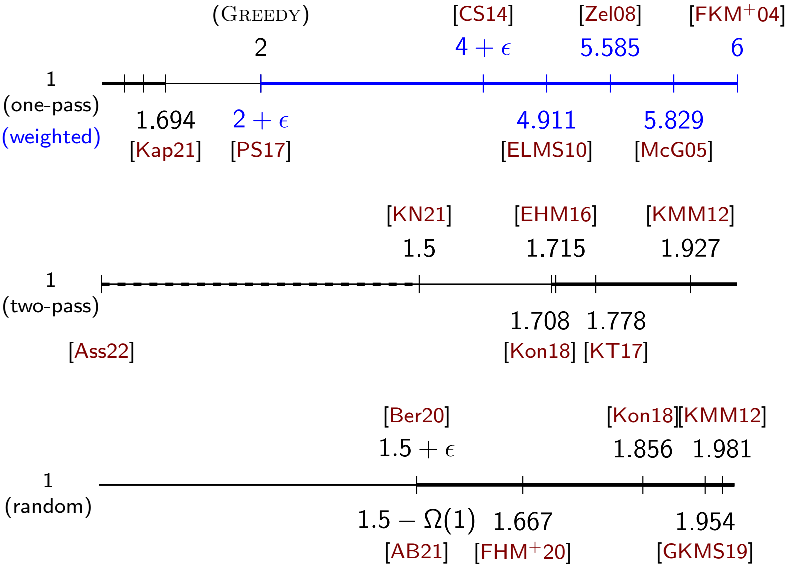
Insertion-Only



Insertion-Only



Insertion-Only



Sliding Window

[CMS13]

$3 + \epsilon$

1

Sliding Window

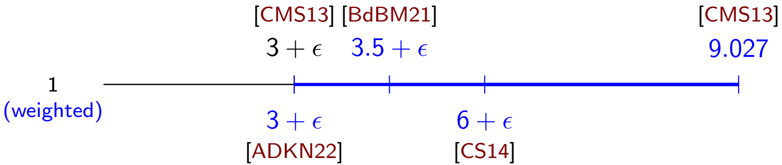
[CMS13]

$3 + \epsilon$

1 ———————————┤

(weighted)

Sliding Window



Dynamic (Insertion-Deletion)

Goal:

- Find an α-approximation

Dynamic (Insertion-Deletion)

Goal:

- Find an α-approximation

Dynamic (Insertion-Deletion)

Goal:

- Find an α-approximation

[AKLY16]
$\Omega(\frac{n^{2-o(1)}}{\alpha^3})$

[Kon15]
$O(\frac{n^2}{\alpha^2} \log^4 n)$

$\Omega(\frac{n^{1.5}}{\alpha^4})$
[Kon15]

$\Omega(\frac{n^2}{\alpha^3})$
[DK20]

$O(\frac{n^2}{\alpha^3} \log^4 n)$
[CCE$^+$16, AKLY16]

Dynamic (Insertion-Deletion)

Goal:

- Find an α-approximation

$\Omega(\frac{n^{1.5}}{\alpha^4})$ [Kon15]

[AKLY16] $\Omega(\frac{n^{2-o(1)}}{\alpha^3})$

[AS22] $O(\frac{n^2}{\alpha^3})$

$\Omega(\frac{n^2}{\alpha^3})$ [DK20]

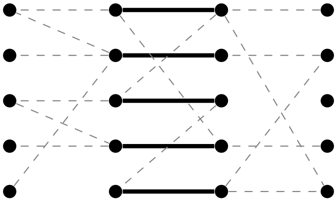
$O(\frac{n^2}{\alpha^3}\log^4 n)$ [CCE+16, AKLY16]

[Kon15] $O(\frac{n^2}{\alpha^2}\log^4 n)$

Relation to Vertex Cover

GREEDY Matching:

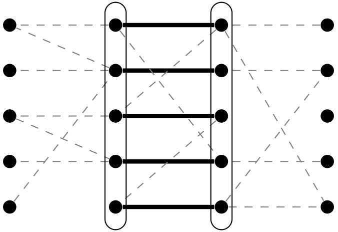
- Maximal
- 2-approximation



Relation to Vertex Cover

GREEDY Vertex Cover:

- 2-approximation



Relation to Vertex Cover (Results)

	matching	**vertex cover**
insertion-only	[1.694, 2]	
sliding window	$[1, 3+\epsilon]$	
dynamic	$\Theta(\frac{n^2}{\alpha^3})$	

Relation to Vertex Cover (Results)

	matching	vertex cover
insertion-only	[1.694, 2]	[1, 2] GREEDY
sliding window	$[1,\ 3+\epsilon]$	
dynamic	$\Theta(\frac{n^2}{\alpha^3})$	

Relation to Vertex Cover (Results)

	matching	**vertex cover**
insertion-only	$[1.694,\ 2]$	$[1,\ 2]$ GREEDY
sliding window	$[1,\ 3+\epsilon]$	$[1,\ 3+\epsilon]$ [Sub21]
dynamic	$\Theta(\frac{n^2}{\alpha^3})$	

Relation to Vertex Cover (Results)

	matching	vertex cover
insertion-only	[1.694, 2]	[1, 2] GREEDY
sliding window	$[1, 3 + \epsilon]$	$[1, 3 + \epsilon]$ [Sub21]
dynamic	$\Theta(\frac{n^2}{\alpha^3})$	$\Theta(\frac{n^2}{\alpha^2})$ [DK20] & this work

Relation to Vertex Cover (Results)

	matching	**vertex cover**
insertion-only	[1.694, 2]	[1, 2] GREEDY
sliding window	$[1, 3+\epsilon]$	$[1, 3+\epsilon]$ [Sub21]
dynamic	$\Theta(\frac{n^2}{\alpha^3})$	$\Theta(\frac{n^2}{\alpha^2})$ [DK20] & this work

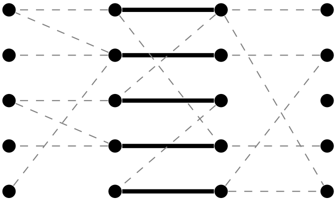
All UBs use GREEDY in some way!

What makes dynamic graph streams hard?

What makes dynamic graph streams hard?

GREEDY Matching:

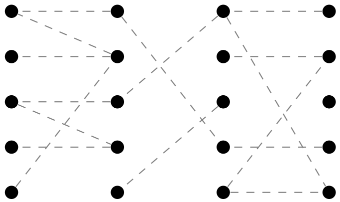
- Maximal
- 2-approximation



What makes dynamic graph streams hard?

GREEDY Matching:

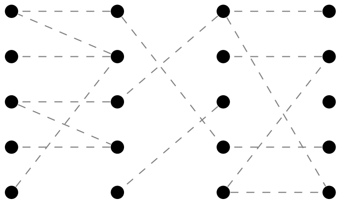
- **Not** Maximal
- **0**-approximation



What makes dynamic graph streams hard?

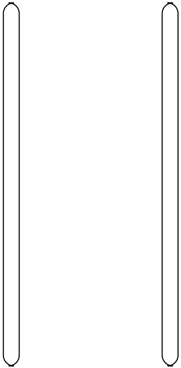
GREEDY Matching:

- **Not** Maximal
- **0**-approximation

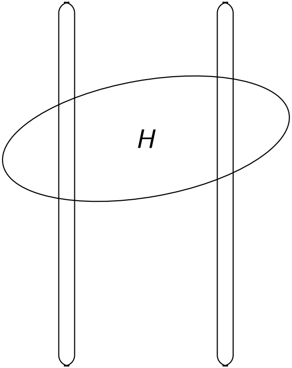


Note: Deterministically returning a single edge requires $\Omega(n^2)$ bits of space for dense graphs.

What techniques are used?

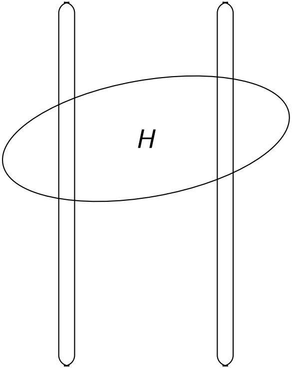


What techniques are used?



For a subgraph H of G with m edges:

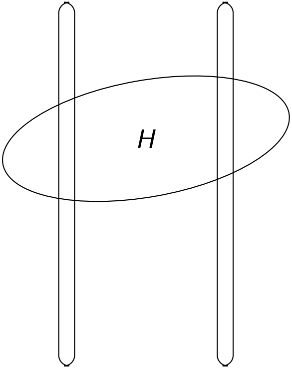
What techniques are used?



For a subgraph H of G with m edges:

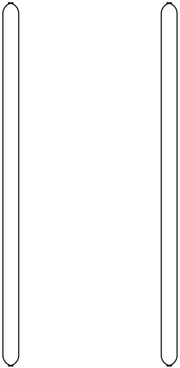
1. Check if H is empty
   - A counter using $\Theta(\log m)$ bits

What techniques are used?

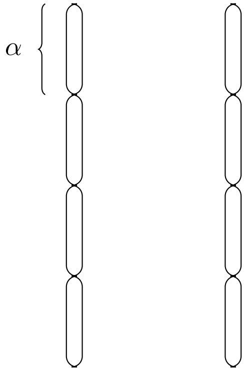


For a subgraph H of G with m edges:

1. Check if H is empty
   - A counter using $\Theta(\log m)$ bits
2. Retrieve an edge if one is present
   - An L_0-sampler using $\Theta(\log^3 n)$ bits
   - Neighbourhood sampler

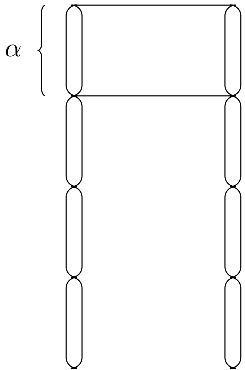


α-Approx Det. Dynamic Vertex Cover [DK20]



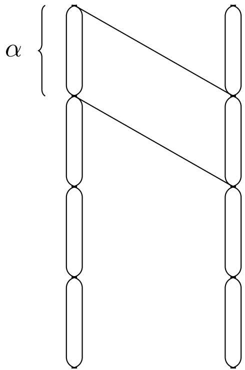
1. Vertex groups of size α
   - about $\frac{n}{\alpha}$ groups

α-Approx Det. Dynamic Vertex Cover [DK20]



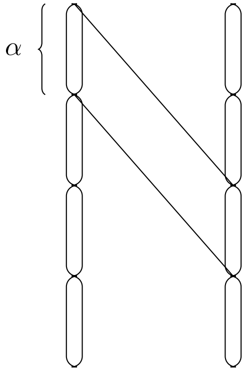
1. Vertex groups of size α
   - about $\frac{n}{\alpha}$ groups
2. Check if there is an edge between each pair of groups
   - about $\frac{n^2}{\alpha^2}$ pairs

α-Approx Det. Dynamic Vertex Cover [DK20]



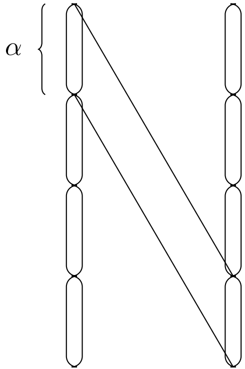
1. Vertex groups of size α
   - about $\frac{n}{\alpha}$ groups
2. Check if there is an edge between each pair of groups
   - about $\frac{n^2}{\alpha^2}$ pairs

α-Approx Det. Dynamic Vertex Cover [DK20]



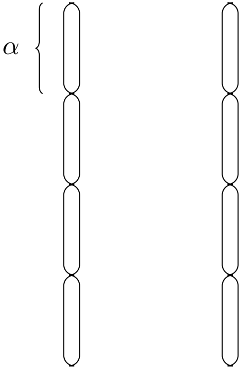
1. Vertex groups of size α
   - about $\frac{n}{\alpha}$ groups
2. Check if there is an edge between each pair of groups
   - about $\frac{n^2}{\alpha^2}$ pairs

α-Approx Det. Dynamic Vertex Cover [DK20]



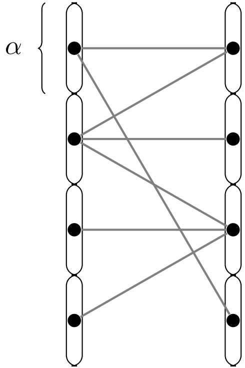
1. Vertex groups of size α
   - about $\frac{n}{\alpha}$ groups
2. Check if there is an edge between each pair of groups
   - about $\frac{n^2}{\alpha^2}$ pairs

α-Approx Det. Dynamic Vertex Cover [DK20]



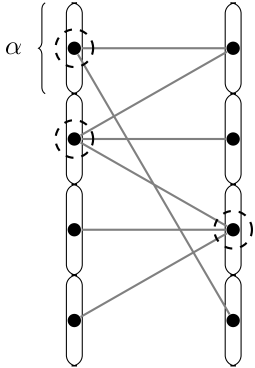
1. Vertex groups of size α
   - about $\frac{n}{\alpha}$ groups
2. Check if there is an edge between each pair of groups
   - about $\frac{n^2}{\alpha^2}$ pairs

α-Approx Det. Dynamic Vertex Cover [DK20]



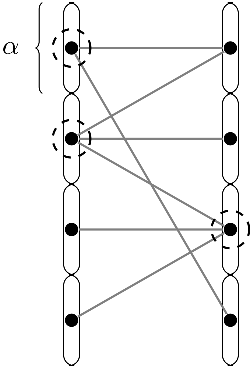
1. Vertex groups of size α
   - about $\frac{n}{\alpha}$ groups
2. Check if there is an edge between each pair of groups
   - about $\frac{n^2}{\alpha^2}$ pairs

α-Approx Det. Dynamic Vertex Cover [DK20]



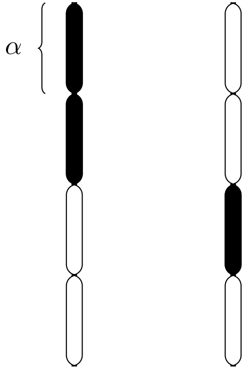
1. Vertex groups of size α
   - about $\frac{n}{\alpha}$ groups
2. Check if there is an edge between each pair of groups
   - about $\frac{n^2}{\alpha^2}$ pairs
3. Return vertices of the group-level vertex cover

α-Approx Det. Dynamic Vertex Cover [DK20]



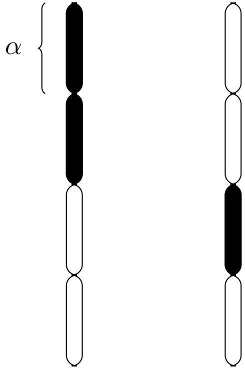
1. Vertex groups of size α
   - about $\frac{n}{\alpha}$ groups
2. Check if there is an edge between each pair of groups
   - about $\frac{n^2}{\alpha^2}$ pairs
3. Return vertices of the group-level vertex cover

α-Approx Det. Dynamic Vertex Cover [DK20]



1. Vertex groups of size α
   - about $\frac{n}{\alpha}$ groups
2. Check if there is an edge between each pair of groups
   - about $\frac{n^2}{\alpha^2}$ pairs
3. Return vertices of the group-level vertex cover

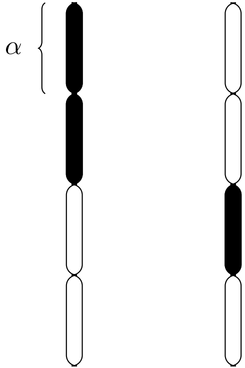
α-Approx Det. Dynamic Vertex Cover [DK20]



1. Vertex groups of size α
   - about $\frac{n}{\alpha}$ groups
2. Check if there is an edge between each pair of groups
   - about $\frac{n^2}{\alpha^2}$ pairs
3. Return vertices of the group-level vertex cover

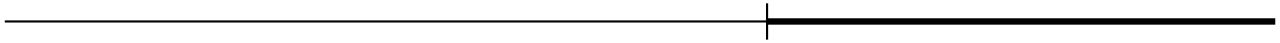
Space: $O(\frac{n^2}{\alpha^2})$ counters, each using $O(\log \alpha)$ bits. Hence, $O(\frac{n^2}{\alpha^2} \log \alpha)$ bits.

α-Approx Det. Dynamic Vertex Cover [DK20]

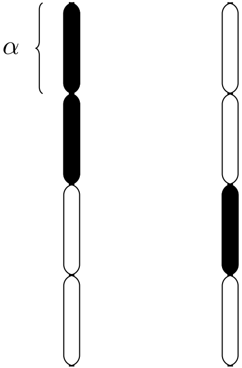


1. Vertex groups of size α
   - about $\frac{n}{\alpha}$ groups
2. Check if there is an edge between each pair of groups
   - about $\frac{n^2}{\alpha^2}$ pairs
3. Return vertices of the group-level vertex cover

Space: $O(\frac{n^2}{\alpha^2})$ counters, each using $O(\log \alpha)$ bits. Hence, $O(\frac{n^2}{\alpha^2} \log \alpha)$ bits.

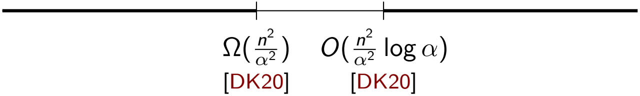


α-Approx Det. Dynamic Vertex Cover [DK20]

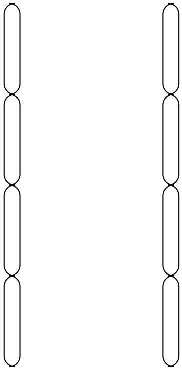


1. Vertex groups of size α
   - about $\frac{n}{\alpha}$ groups
2. Check if there is an edge between each pair of groups
   - about $\frac{n^2}{\alpha^2}$ pairs
3. Return vertices of the group-level vertex cover

Space: $O(\frac{n^2}{\alpha^2})$ counters, each using $O(\log \alpha)$ bits. Hence, $O(\frac{n^2}{\alpha^2} \log \alpha)$ bits.



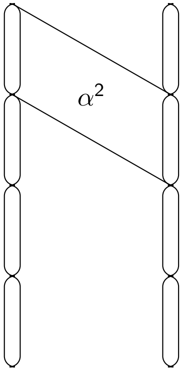
What's the issue?



What's the issue?

Problem:

- Counters use $O(\log \alpha)$ bits.
- Each counter counts upto α^2 edges.



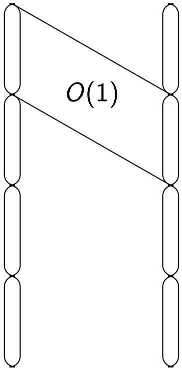
What's the issue?

Problem:

- Counters use $O(\log \alpha)$ bits.
- Each counter counts upto α^2 edges.

Goal:

- Counters to use $O(1)$ bits.
- Counters to count upto constant many edges



What's the issue?

Problem:

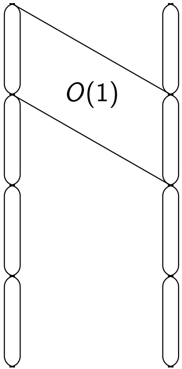
- Counters use $O(\log \alpha)$ bits.
- Each counter counts upto α^2 edges.

Goal:

- Counters to use $O(1)$ bits.
- Counters to count upto constant many edges

How?

- Sparse graph
- Randomly partition



What's the issue?

Problem:

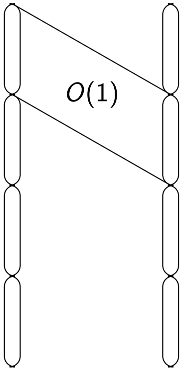
- Counters use $O(\log \alpha)$ bits.
- Each counter counts upto α^2 edges.

Goal:

- Counters to use $O(1)$ bits.
- Counters to count upto constant many edges

How?

- Sparse graph
- Randomly partition (easy)



What's the issue?

Problem:

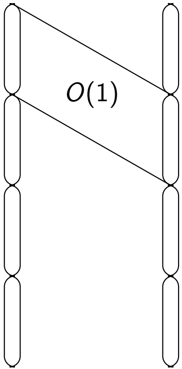
- Counters use $O(\log \alpha)$ bits.
- Each counter counts upto α^2 edges.

Goal:

- Counters to use $O(1)$ bits.
- Counters to count upto constant many edges

How?

- Sparse graph (GREEDY!)
- Randomly partition (easy)



Sparsness properties of GREEDY

Lemma

Let G be a n-vertex graph with m edges and let M_s be a GREEDY *matching on $s \leq m$ uniform randomly sampled edges. Then, for $G_R = G[V \backslash V(M_s)]$,*

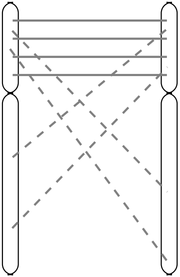
$$\Delta(G_R) \leq \frac{C \cdot m \cdot \log n}{s}.$$

Sparsness properties of GREEDY

Lemma

Let G be a n-vertex graph with m edges and let M_s be a GREEDY *matching on $s \leq m$ uniform randomly sampled edges. Then, for $G_R = G[V \backslash V(M_s)]$,*

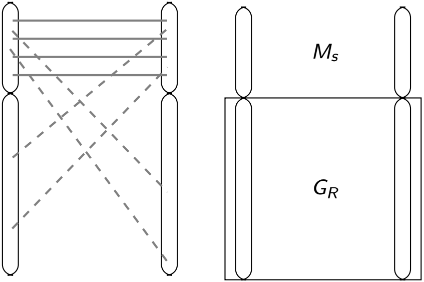
$$\Delta(G_R) \leq \frac{C \cdot m \cdot \log n}{s}.$$



Lemma

Let G be a n-vertex graph with m edges and let M_s be a GREEDY *matching on $s \leq m$ uniform randomly sampled edges. Then, for $G_R = G[V \backslash V(M_s)]$,*

$$\Delta(G_R) \leq \frac{C \cdot m \cdot \log n}{s}.$$

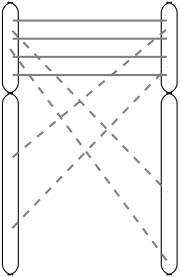


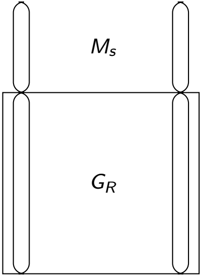
Sparsness properties of Greedy

Lemma

Let G be a n-vertex graph with m edges and let M_s be a Greedy *matching on $s \leq m$ uniform randomly sampled edges. Then, for $G_R = G[V \backslash V(M_s)]$,*

$$\Delta(G_R) \leq \frac{C \cdot m \cdot \log n}{s}.$$





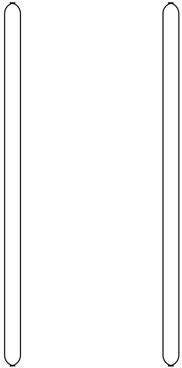
Proof (sketch).

For any vertex $v \in G$,

- $v \in G_R$
- $\deg_{G_R}(v) > \frac{C \cdot m \cdot \log n}{s}$

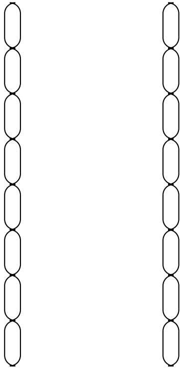
both do not occur w.h.p. □

Algorithm

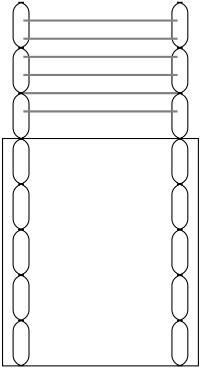


Algorithm

1. Randomly partition vertices ($\frac{n}{\alpha}$ groups)

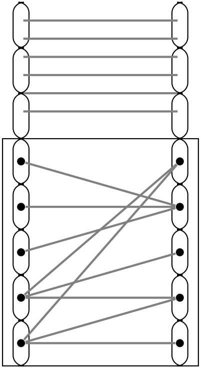


1. Randomly partition vertices ($\frac{n}{\alpha}$ groups)
2. Sample $O(\frac{n^2}{\alpha^2 \log^3 n})$ edges randomly and run GREEDY on them to get M



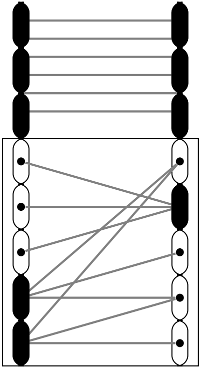
Algorithm

1. Randomly partition vertices ($\frac{n}{\alpha}$ groups)
2. Sample $O(\frac{n^2}{\alpha^2 \log^3 n})$ edges randomly and run GREEDY on them to get M
3. Check if an edge is present between pairs and compute group-level vertex cover



Algorithm

1. Randomly partition vertices ($\frac{n}{\alpha}$ groups)
2. Sample $O(\frac{n^2}{\alpha^2 \log^3 n})$ edges randomly and run GREEDY on them to get M
3. Check if an edge is present between pairs and compute group-level vertex cover
4. Return vertices of the covering groups including those with matched vertices



Limitation:

- Spareseness properties are only sufficient for $\alpha \ll n^{\frac{1}{3.5}}$

Limitation:

- Spareseness properties are only sufficient for $\alpha \ll n^{\frac{1}{3.5}}$

Extension:

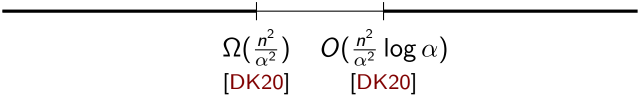
- Non-uniform sampling instead of uniform sampling, i.e., using neighbourhood edge sampling [AS22]
- Results are an average degree bound instead of max degree bound
- Full range, i.e., $\alpha \ll n$

Limitation:

- Spareseness properties are only sufficient for $\alpha \ll n^{\frac{1}{3.5}}$

Extension:

- Non-uniform sampling instead of uniform sampling, i.e., using neighbourhood edge sampling [AS22]
- Results are an average degree bound instead of max degree bound
- Full range, i.e., $\alpha \ll n$



Limitation:

- Spareseness properties are only sufficient for $\alpha \ll n^{\frac{1}{3.5}}$

Extension:

- Non-uniform sampling instead of uniform sampling, i.e., using neighbourhood edge sampling [AS22]
- Results are an average degree bound instead of max degree bound
- Full range, i.e., $\alpha \ll n$

[this work]

$O(\frac{n^2}{\alpha^2})$

$\Omega(\frac{n^2}{\alpha^2})$ [DK20]

$O(\frac{n^2}{\alpha^2} \log \alpha)$ [DK20]

Limitation:

- Spareseness properties are only sufficient for $\alpha \ll n^{\frac{1}{3.5}}$

Extension:

- Non-uniform sampling instead of uniform sampling, i.e., using neighbourhood edge sampling [AS22]
- Results are an average degree bound instead of max degree bound
- Full range, i.e., $\alpha \ll n$

[this work]

$O(\frac{n^2}{\alpha^2})$

$\Omega(\frac{n^2}{\alpha^2})$ [DK20] $\quad O(\frac{n^2}{\alpha^2} \log \alpha)$ [DK20]

Open Questions:

- Deterministic techniques or LB instead
- Other problems like dominating set and spectral sparsification

Thank You

Questions?

References I

Sepehr Assadi and Soheil Behnezhad, *Beating two-thirds for random-order streaming matching*, 48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference) (Nikhil Bansal, Emanuela Merelli, and James Worrell, eds.), LIPIcs, vol. 198, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 19:1–19:13.

Cezar-Mihail Alexandru, Pavel Dvorák, Christian Konrad, and Kheeran K. Naidu, *Closing the gap between weighted and unweighted matching in the sliding window model*, CoRR **abs/2204.04717** (2022).

References II

Kook Jin Ahn, Sudipto Guha, and Andrew McGregor, *Analyzing graph structure via linear measurements*, Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012 (Yuval Rabani, ed.), SIAM, 2012, pp. 459–467.

Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev, *Maximum matchings in dynamic graph streams and the simultaneous communication model*, Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016 (Robert Krauthgamer, ed.), SIAM, 2016, pp. 1345–1364.

References III

- Sepehr Assadi and Vihan Shah, *An asymptotically optimal algorithm for maximum matching in dynamic streams*, 13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA (Mark Braverman, ed.), LIPIcs, vol. 215, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, pp. 9:1–9:23.

- Sepehr Assadi, *A two-pass (conditional) lower bound for semi-streaming maximum matching*, Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022 (Joseph (Seffi) Naor and Niv Buchbinder, eds.), SIAM, 2022, pp. 708–742.

References IV

- Leyla Biabani, Mark de Berg, and Morteza Monemizadeh, *Maximum-weight matching in sliding windows and beyond*, 32nd International Symposium on Algorithms and Computation, ISAAC 2021, December 6-8, 2021, Fukuoka, Japan (Hee-Kap Ahn and Kunihiko Sadakane, eds.), LIPIcs, vol. 212, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 73:1–73:16.
- Aaron Bernstein, *Improved bounds for matching in random-order streams*, 47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference) (Artur Czumaj, Anuj Dawar, and Emanuela Merelli, eds.), LIPIcs, vol. 168, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, pp. 12:1–12:13.

References V

Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova, *Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams*, Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016 (Robert Krauthgamer, ed.), SIAM, 2016, pp. 1326–1344.

Michael S. Crouch, Andrew McGregor, and Daniel M. Stubbs, *Dynamic graphs in the sliding-window model*, Algorithms - ESA 2013 - 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings (Hans L. Bodlaender and Giuseppe F. Italiano, eds.), Lecture Notes in Computer Science, vol. 8125, Springer, 2013, pp. 337–348.

References VI

Michael S. Crouch and Daniel M. Stubbs, *Improved streaming algorithms for weighted matching, via unweighted matching*, Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain (Klaus Jansen, José D. P. Rolim, Nikhil R. Devanur, and Cristopher Moore, eds.), LIPIcs, vol. 28, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014, pp. 96–104.

Jacques Dark and Christian Konrad, *Optimal lower bounds for matching and vertex cover in dynamic graph streams*, 35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference) (Shubhangi Saraf, ed.), LIPIcs, vol. 169, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, pp. 30:1–30:14.

References VII

Hossein Esfandiari, MohammadTaghi Hajiaghayi, and Morteza Monemizadeh, *Finding large matchings in semi-streaming*, IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain (Carlotta Domeniconi, Francesco Gullo, Francesco Bonchi, Josep Domingo-Ferrer, Ricardo Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu, eds.), IEEE Computer Society, 2016, pp. 608–614.

Leah Epstein, Asaf Levin, Julián Mestre, and Danny Segev, *Improved approximation guarantees for weighted matching in the semi-streaming model*, 27th International Symposium on Theoretical Aspects of Computer Science, STACS 2010, March 4-6, 2010, Nancy, France (Jean-Yves Marion and Thomas Schwentick, eds.), LIPIcs, vol. 5, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010, pp. 347–358.

Alireza Farhadi, Mohammad Taghi Hajiaghayi, Tung Mai, Anup Rao, and Ryan A. Rossi, *Approximate maximum matching in random streams*, Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020 (Shuchi Chawla, ed.), SIAM, 2020, pp. 1773–1785.

Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang, *On graph problems in a semi-streaming model*, Automata, Languages and Programming: 31st International Colloquium, ICALP 2004, Turku, Finland, July 12-16, 2004. Proceedings (Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella, eds.), Lecture Notes in Computer Science, vol. 3142, Springer, 2004, pp. 531–543.

References IX

Ashish Goel, Michael Kapralov, and Sanjeev Khanna, *On the communication and streaming complexity of maximum bipartite matching*, Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012 (Yuval Rabani, ed.), SIAM, 2012, pp. 468–485.

Buddhima Gamlath, Sagar Kale, Slobodan Mitrovic, and Ola Svensson, *Weighted matchings via unweighted augmentations*, Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019 (Peter Robinson and Faith Ellen, eds.), ACM, 2019, pp. 491–500.

Michael Kapralov, *Better bounds for matchings in the streaming model*, Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013 (Sanjeev Khanna, ed.), SIAM, 2013, pp. 1679–1697.

———, *Space lower bounds for approximating maximum matching in the edge arrival model*, Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021 (Dániel Marx, ed.), SIAM, 2021, pp. 1874–1893.

Christian Konrad, Frédéric Magniez, and Claire Mathieu, *Maximum matching in semi-streaming with few passes*, Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings (Anupam Gupta, Klaus Jansen, José D. P. Rolim, and Rocco A. Servedio, eds.), Lecture Notes in Computer Science, vol. 7408, Springer, 2012, pp. 231–242.

Christian Konrad and Kheeran K. Naidu, *On two-pass streaming algorithms for maximum bipartite matching*, Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference) (Mary Wootters and Laura Sanità, eds.), LIPIcs, vol. 207, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 19:1–19:18.

Christian Konrad, *Maximum matching in turnstile streams*, Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings (Nikhil Bansal and Irene Finocchi, eds.), Lecture Notes in Computer Science, vol. 9294, Springer, 2015, pp. 840–852.

———, *A simple augmentation method for matchings with applications to streaming algorithms*, 43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK (Igor Potapov, Paul G. Spirakis, and James Worrell, eds.), LIPIcs, vol. 117, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, pp. 74:1–74:16.

Sagar Kale and Sumedh Tirodkar, *Maximum matching in two, three, and a few more passes over graph streams*, Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA (Klaus Jansen, José D. P. Rolim, David Williamson, and Santosh S. Vempala, eds.), LIPIcs, vol. 81, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, pp. 15:1–15:21.

Andrew McGregor, *Finding graph matchings in data streams*, Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th InternationalWorkshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings (Chandra Chekuri, Klaus Jansen, José D. P. Rolim, and Luca Trevisan, eds.), Lecture Notes in Computer Science, vol. 3624, Springer, 2005, pp. 170–181.

S. Muthukrishnan, *Data streams: algorithms and applications*, Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA, ACM/SIAM, 2003, pp. 413–413.

Ami Paz and Gregory Schwartzman, *A $2+\epsilon$)-approximation for maximum weight matching in the semi-streaming model*, Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19 (Philip N. Klein, ed.), SIAM, 2017, pp. 2153–2161.

Sai Krishna Chaitanya Nalam Venkata Subrahmanya, *Vertex cover in the sliding window model*, Rutgers University, Retrieved from https://doi.org/doi:10.7282/t3-xyr3-1446 (2021).

Mariano Zelke, *Weighted matching in the semi-streaming model*, STACS 2008, 25th Annual Symposium on Theoretical Aspects of Computer Science, Bordeaux, France, February 21-23, 2008, Proceedings (Susanne Albers and Pascal Weil, eds.), LIPIcs, vol. 1, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2008, pp. 669–680.