Graph Streaming and Maximum Matching (in a Few Passes)

Kheeran K. Naidu

University of Bristol, UK

kheeran.naidu@bristol.ac.uk

Graphs and Bipartite Graphs

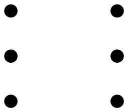
Definition (Graph)

A graph is made up of entities called vertices and relations between them called edges.

Graphs and Bipartite Graphs

Definition (Graph)

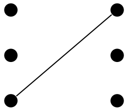
A graph is made up of entities called vertices and relations between them called edges.



Graphs and Bipartite Graphs

Definition (Graph)

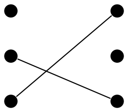
A graph is made up of entities called vertices and relations between them called edges.



Graphs and Bipartite Graphs

Definition (Graph)

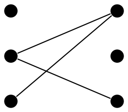
A graph is made up of entities called vertices and relations between them called edges.



Graphs and Bipartite Graphs

Definition (Graph)

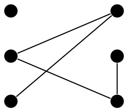
A graph is made up of entities called vertices and relations between them called edges.



Graphs and Bipartite Graphs

Definition (Graph)

A graph is made up of entities called vertices and relations between them called edges.



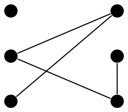
Graphs and Bipartite Graphs

Definition (Graph)

A graph is made up of entities called vertices and relations between them called edges.

Definition (Bipartite Graph)

A bipartite graph $G = (A, B, E)$ is such that, for all $(a, b) \in E$, $a \in A$ and $b \in B$.



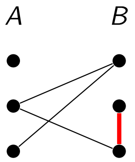
Graphs and Bipartite Graphs

Definition (Graph)

A graph is made up of entities called vertices and relations between them called edges.

Definition (Bipartite Graph)

A bipartite graph $G = (A, B, E)$ is such that, for all $(a, b) \in E$, $a \in A$ and $b \in B$.



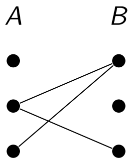
Graphs and Bipartite Graphs

Definition (Graph)

A graph is made up of entities called vertices and relations between them called edges.

Definition (Bipartite Graph)

A bipartite graph $G = (A, B, E)$ is such that, for all $(a, b) \in E$, $a \in A$ and $b \in B$.



Graphs and Bipartite Graphs

Definition (Graph)

A graph is made up of entities called vertices and relations between them called edges.

Definition (Bipartite Graph)

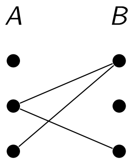
A bipartite graph $G = (A, B, E)$ is such that, for all $(a, b) \in E$, $a \in A$ and $b \in B$.

Observation

For any $G = (A, B, E)$ such that $|A| = |B| = n/2$,

$$0 \leq |E| \leq n^2/4,$$

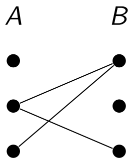
that is, graph G has $O(n^2)$ edges.



Graphs and Bipartite Graphs

Definition (Graph)

A graph is made up of entities called vertices and relations between them called edges.



Definition (Bipartite Graph)

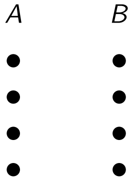
A bipartite graph $G = (A, B, E)$ is such that, for all $(a, b) \in E$, $a \in A$ and $b \in B$.

Observation

For any $G = (A, B, E)$ such that $|A| = |B| = n/2$,

$$0 \leq |E| \leq n^2/4,$$

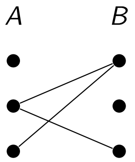
that is, graph G has $O(n^2)$ edges.



Graphs and Bipartite Graphs

Definition (Graph)

A graph is made up of entities called vertices and relations between them called edges.



Definition (Bipartite Graph)

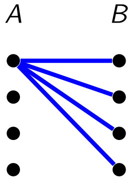
A bipartite graph $G = (A, B, E)$ is such that, for all $(a, b) \in E$, $a \in A$ and $b \in B$.

Observation

For any $G = (A, B, E)$ such that $|A| = |B| = n/2$,

$$0 \leq |E| \leq n^2/4,$$

that is, graph G has $O(n^2)$ edges.



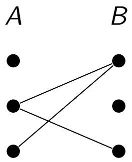
Graphs and Bipartite Graphs

Definition (Graph)

A graph is made up of entities called vertices and relations between them called edges.

Definition (Bipartite Graph)

A bipartite graph $G = (A, B, E)$ is such that, for all $(a, b) \in E$, $a \in A$ and $b \in B$.

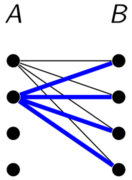


Observation

For any $G = (A, B, E)$ such that $|A| = |B| = n/2$,

$$0 \leq |E| \leq n^2/4,$$

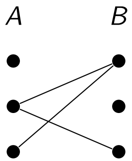
that is, graph G has $O(n^2)$ edges.



Graphs and Bipartite Graphs

Definition (Graph)

A graph is made up of entities called vertices and relations between them called edges.



Definition (Bipartite Graph)

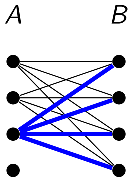
A bipartite graph $G = (A, B, E)$ is such that, for all $(a, b) \in E$, $a \in A$ and $b \in B$.

Observation

For any $G = (A, B, E)$ such that $|A| = |B| = n/2$,

$$0 \leq |E| \leq n^2/4,$$

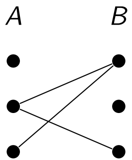
that is, graph G has $O(n^2)$ edges.



Graphs and Bipartite Graphs

Definition (Graph)

A graph is made up of entities called vertices and relations between them called edges.



Definition (Bipartite Graph)

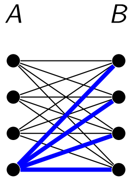
A bipartite graph $G = (A, B, E)$ is such that, for all $(a, b) \in E$, $a \in A$ and $b \in B$.

Observation

For any $G = (A, B, E)$ such that $|A| = |B| = n/2$,

$$0 \leq |E| \leq n^2/4,$$

that is, graph G has $O(n^2)$ edges.



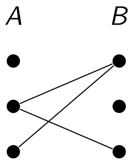
Graphs and Bipartite Graphs

Definition (Graph)

A graph is made up of entities called vertices and relations between them called edges.

Definition (Bipartite Graph)

A bipartite graph $G = (A, B, E)$ is such that, for all $(a, b) \in E$, $a \in A$ and $b \in B$.

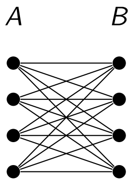


Observation

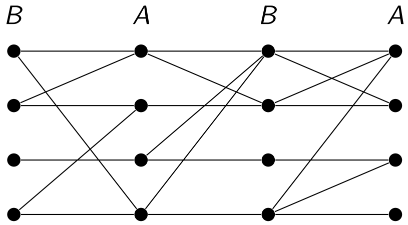
For any $G = (A, B, E)$ such that $|A| = |B| = n/2$,

$$0 \leq |E| \leq n^2/4,$$

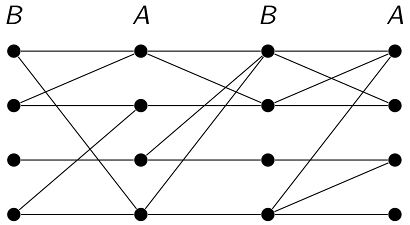
that is, graph G has $O(n^2)$ edges.



Maximum Bipartite Matching (MBM)



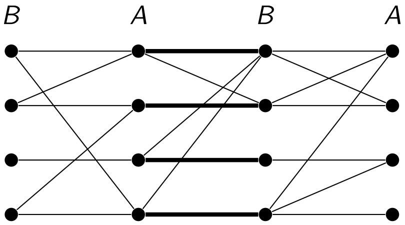
Maximum Bipartite Matching (MBM)



Matchings

A matching M is a subset of vertex-disjoint edges of a graph.

Maximum Bipartite Matching (MBM)

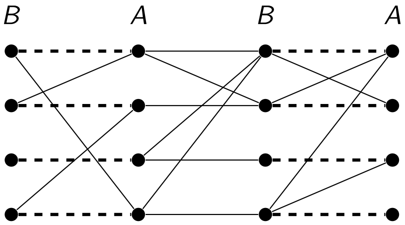


Matchings

A matching M is a subset of vertex-disjoint edges of a graph.

- **Maximal**: Every edge $e \in E \setminus M$ is incident to M.

Maximum Bipartite Matching (MBM)

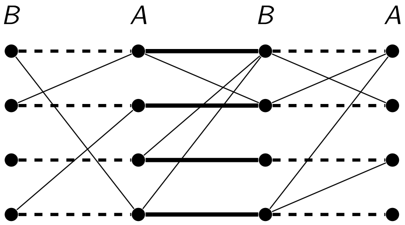


Matchings

A matching M is a subset of vertex-disjoint edges of a graph.

- **Maximal**: Every edge $e \in E \setminus M$ is incident to M.
- **Maximum**: Largest size, $\mu(G)$.

Maximum Bipartite Matching (MBM)



Matchings

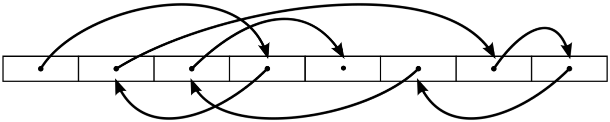
A matching M is a subset of vertex-disjoint edges of a graph.

- **Maximal**: Every edge $e \in E \setminus M$ is incident to M.
- **Maximum**: Largest size, $\mu(G)$.
- Maximal matchings are 0.5-approximations of maximum matchings.

Traditional Model of Computation

Assumption

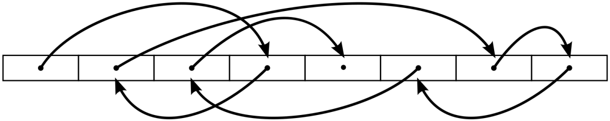
Algorithms have random access to the entire graph.



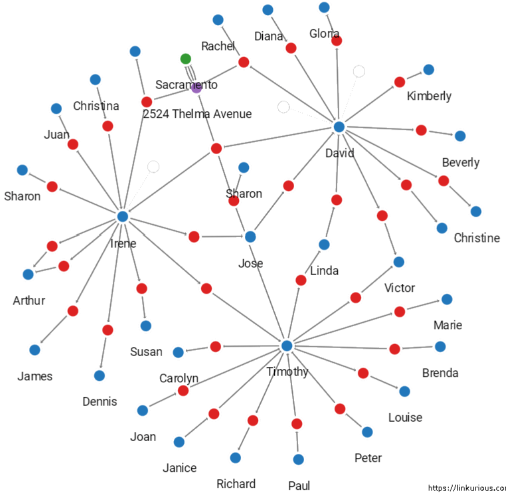
Traditional Model of Computation

Assumption

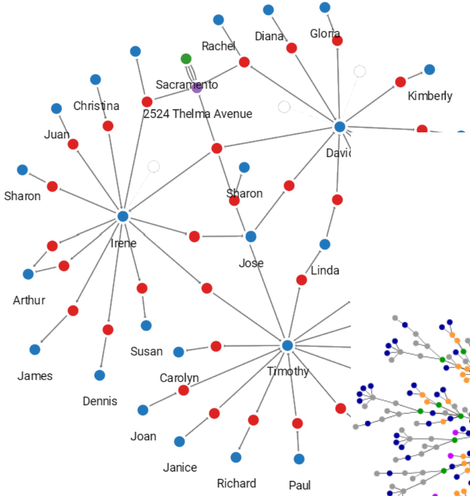
Algorithms have random access to the entire graph.

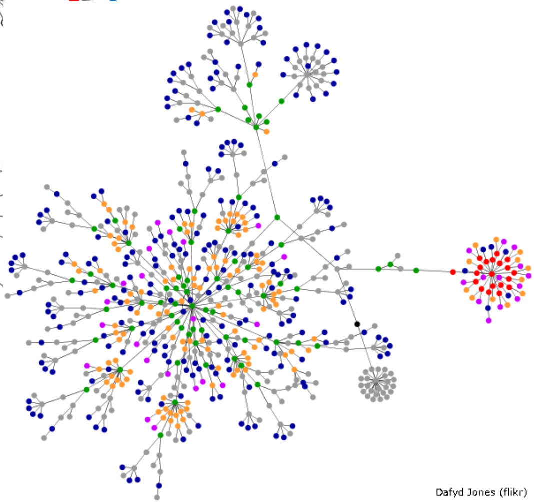


What if the input is extremely large?

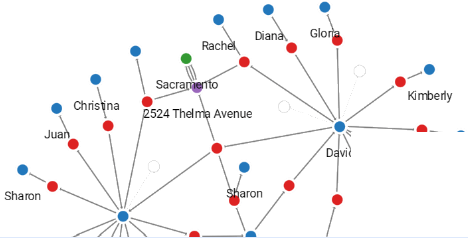


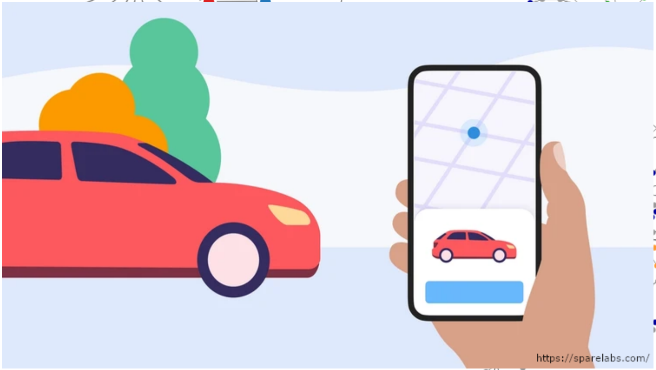
https://linkurious.com/



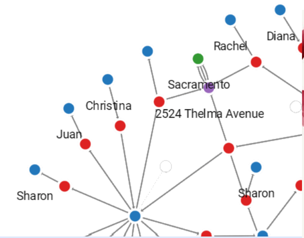


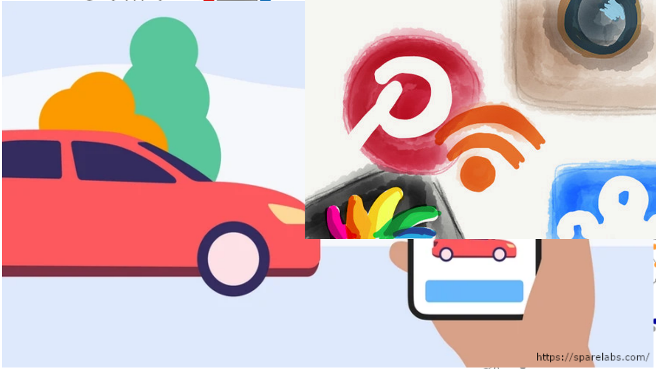
Dafyd Jones (flickr)

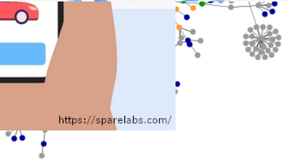




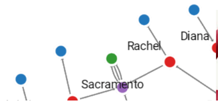
Dafyd Jones (flikr)

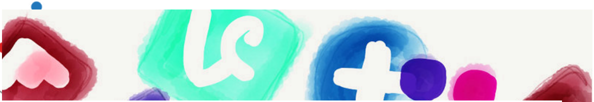




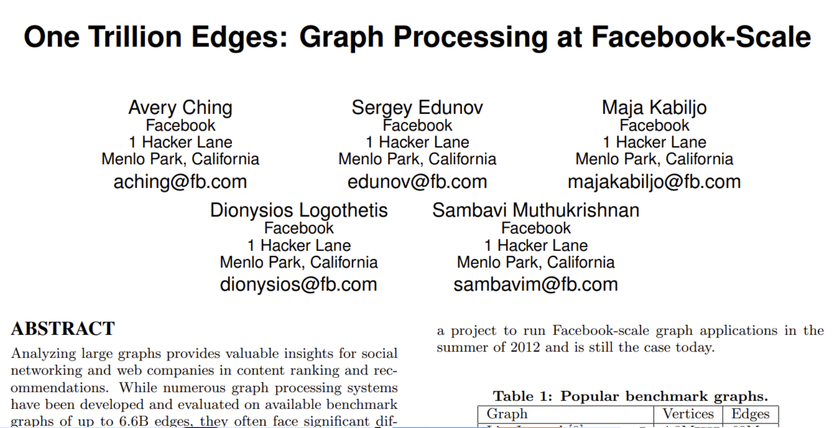


Dafyd Jones (flikr)





(flikr)



One Trillion Edges: Graph Processing at Facebook-Scale

Avery Ching
Facebook
1 Hacker Lane
Menlo Park, California
aching@fb.com

Sergey Edunov
Facebook
1 Hacker Lane
Menlo Park, California
edunov@fb.com

Maja Kabiljo
Facebook
1 Hacker Lane
Menlo Park, California
majakabiljo@fb.com

Dionysios Logothetis
Facebook
1 Hacker Lane
Menlo Park, California
dionysios@fb.com

Sambavi Muthukrishnan
Facebook
1 Hacker Lane
Menlo Park, California
sambavim@fb.com

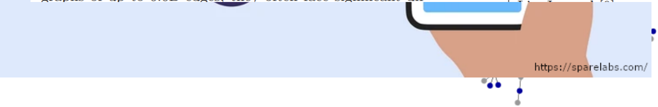
ABSTRACT

Analyzing large graphs provides valuable insights for social networking and web companies in content ranking and recommendations. While numerous graph processing systems have been developed and evaluated on available benchmark graphs of up to 6.6B edges, they often face significant dif-

a project to run Facebook-scale graph applications in the summer of 2012 and is still the case today.

Table 1: Popular benchmark graphs.

Graph	Vertices	Edges



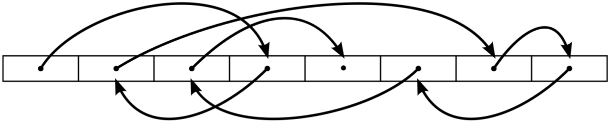
https://sparelabs.com/

Dafyd Jones (flikr)

Massive Graphs

Assumption (Infeasible)

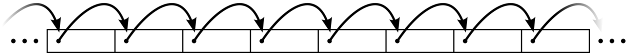
Algorithms have random access to the entire graph.



Graph Streaming (Model of Computation)

Definition

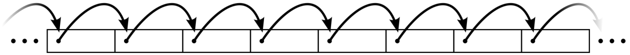
An n-vertex graph is presented as a sequence of edges to an algorithm.



Graph Streaming (Model of Computation)

Definition

An n-vertex graph is presented as a sequence of edges to an algorithm.

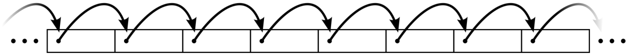


Insertion-only [FKM+04]

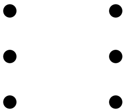
Graph Streaming (Model of Computation)

Definition

An *n*-vertex graph is presented as a sequence of edges to an algorithm.



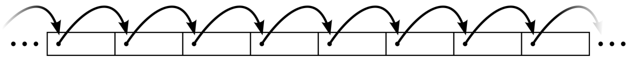
Insertion-only [FKM+04]



Graph Streaming (Model of Computation)

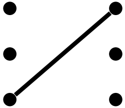
Definition

An *n*-vertex graph is presented as a sequence of edges to an algorithm.



Insertion-only [FKM$^+$04]

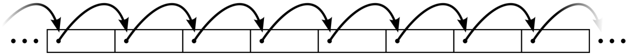
$\boxed{e_1}$



Graph Streaming (Model of Computation)

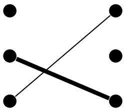
Definition

An n-vertex graph is presented as a sequence of edges to an algorithm.



Insertion-only [FKM+04]

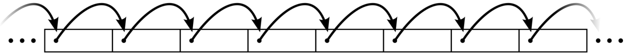
e_1	e_2



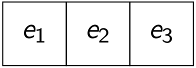
Graph Streaming (Model of Computation)

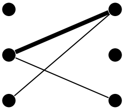
Definition

An *n*-vertex graph is presented as a sequence of edges to an algorithm.



Insertion-only [FKM+04]

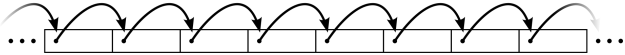




Graph Streaming (Model of Computation)

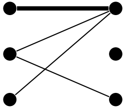
Definition

An *n*-vertex graph is presented as a sequence of edges to an algorithm.



Insertion-only [FKM+04]

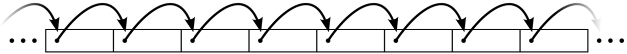
e_1	e_2	e_3	e_4



Graph Streaming (Model of Computation)

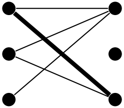
Definition

An *n*-vertex graph is presented as a sequence of edges to an algorithm.



Insertion-only [FKM+04]

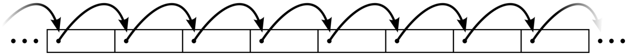
e_1	e_2	e_3	e_4	e_5



Graph Streaming (Model of Computation)

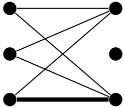
Definition

An *n-vertex graph* is presented as a sequence of edges to an algorithm.



Insertion-only [FKM+04]

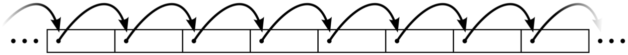
e_1	e_2	e_3	e_4	e_5	e_6



Graph Streaming (Model of Computation)

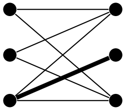
Definition

An n-vertex graph is presented as a sequence of edges to an algorithm.



Insertion-only [FKM+04]

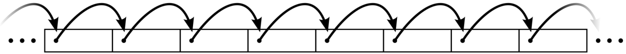
e_1	e_2	e_3	e_4	e_5	e_6	e_7



Graph Streaming (Model of Computation)

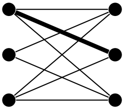
Definition

An *n-vertex graph* is presented as a sequence of edges to an algorithm.



Insertion-only [FKM+04]

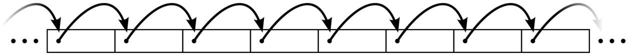
e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



Graph Streaming (Model of Computation)

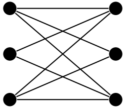
Definition

An *n*-vertex graph is presented as a sequence of edges to an algorithm.



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

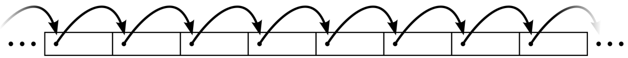
e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



Graph Streaming (Model of Computation)

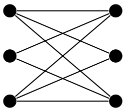
Definition

An n-vertex graph is presented as a sequence of edges to an algorithm.



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

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



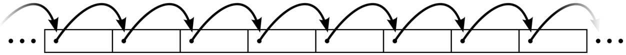
Trivial Algorithm

- Stores all edges using $O(n^2)$ space.

Graph Streaming (Model of Computation)

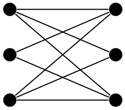
Definition

An *n-vertex graph* is presented as a sequence of edges to an algorithm.



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

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8



Trivial Algorithm

- Stores all edges using $O(n^2)$ space.

Interesting Algorithm

- Uses $O(n \operatorname{polylog} n)$ space.
- Uses one or more passes.

GREEDY: Simple and Powerful

Greedy: Simple and Powerful

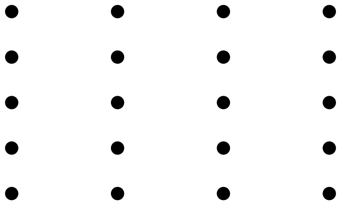
Greedy Matching:

1. Add edge if neither endpoint is matched

GREEDY: Simple and Powerful

GREEDY Matching:

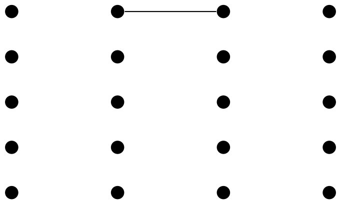
1. Add edge if neither endpoint is matched



GREEDY: Simple and Powerful

GREEDY Matching:

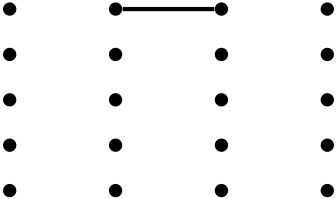
1. Add edge if neither endpoint is matched



GREEDY: Simple and Powerful

GREEDY Matching:

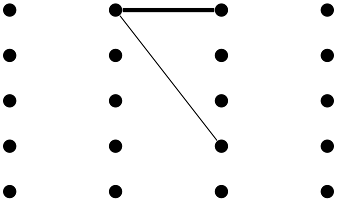
1. Add edge if neither endpoint is matched



Greedy: Simple and Powerful

Greedy Matching:

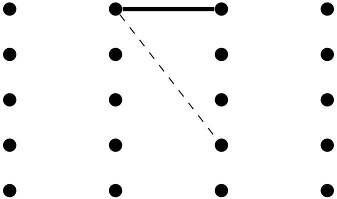
1. Add edge if neither endpoint is matched



GREEDY: Simple and Powerful

GREEDY Matching:

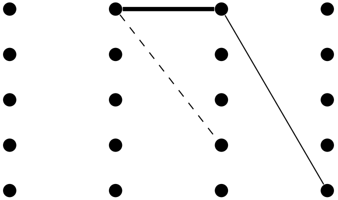
1. Add edge if neither endpoint is matched



GREEDY: Simple and Powerful

GREEDY Matching:

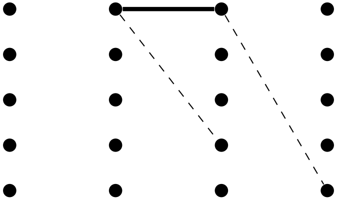
1. Add edge if neither endpoint is matched



GREEDY: Simple and Powerful

GREEDY Matching:

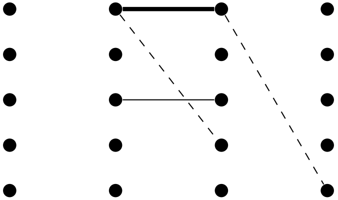
1. Add edge if neither endpoint is matched



GREEDY: Simple and Powerful

GREEDY Matching:

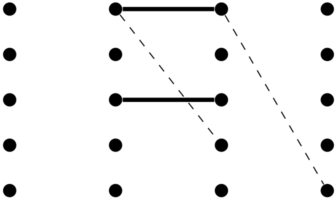
1. Add edge if neither endpoint is matched



GREEDY: Simple and Powerful

GREEDY Matching:

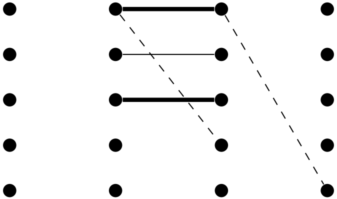
1. Add edge if neither endpoint is matched



GREEDY: Simple and Powerful

GREEDY Matching:

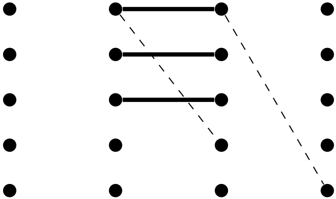
1. Add edge if neither endpoint is matched



Greedy: Simple and Powerful

Greedy Matching:

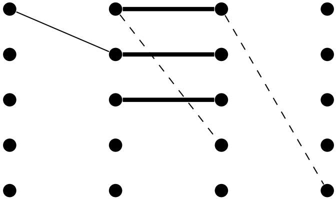
1. Add edge if neither endpoint is matched



Greedy: Simple and Powerful

Greedy Matching:

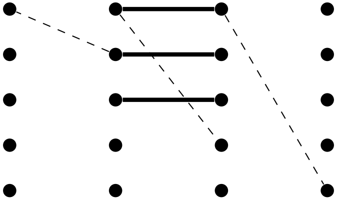
1. Add edge if neither endpoint is matched



GREEDY: Simple and Powerful

GREEDY Matching:

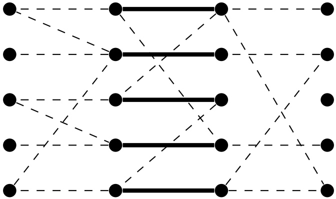
1. Add edge if neither endpoint is matched



Greedy: Simple and Powerful

Greedy Matching:

1. Add edge if neither endpoint is matched

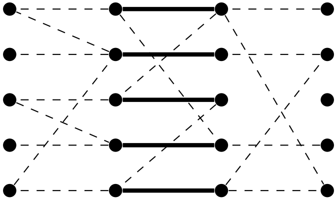


GREEDY: Simple and Powerful

GREEDY Matching:

1. Add edge if neither endpoint is matched

- Maximal

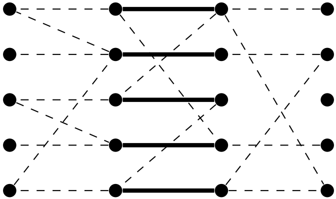


GREEDY: Simple and Powerful

GREEDY Matching:

1. Add edge if neither endpoint is matched

- Maximal
- 0.5-approximation

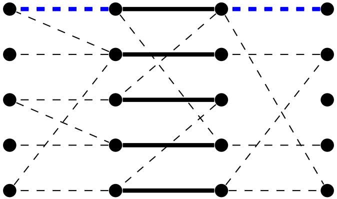


GREEDY: Simple and Powerful

GREEDY Matching:

1. Add edge if neither endpoint is matched

- Maximal
- 0.5-approximation

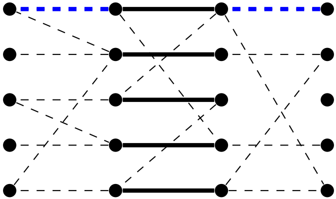


GREEDY: Simple and Powerful

GREEDY Matching:

1. Add edge if neither endpoint is matched

- Maximal
- 0.5-approximation



Space Used

Only the matching M is stored by the algorithm $\implies$ $O(n \log n)$ space since each edge requires $\Theta(\log n)$ space to store.

Approximate MBM using $O(n \operatorname{polylog} n)$ Space

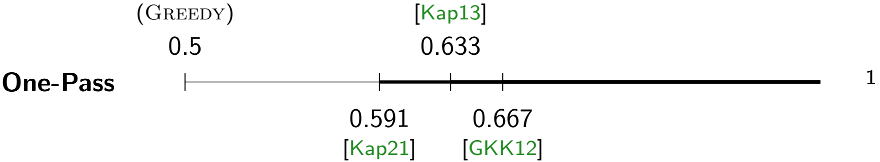
Approximate MBM using $O(n \operatorname{polylog} n)$ Space

(Greedy)

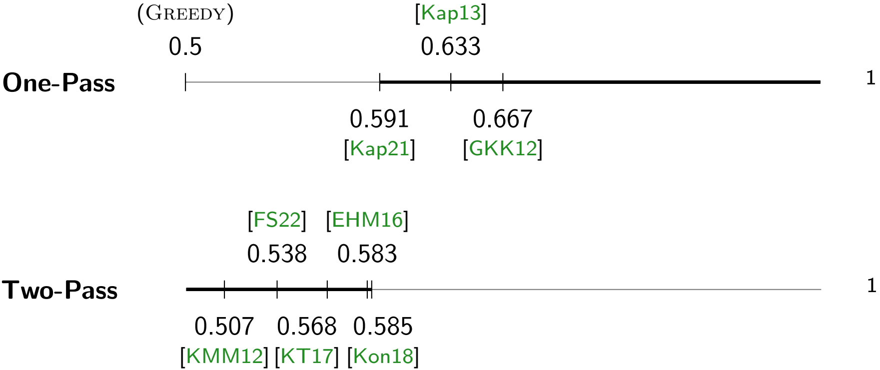
0.5

One-Pass 1

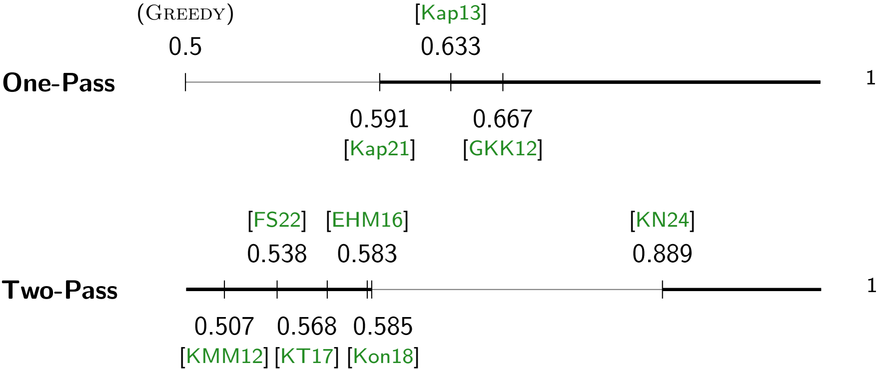
Approximate MBM using $O(n \operatorname{polylog} n)$ Space



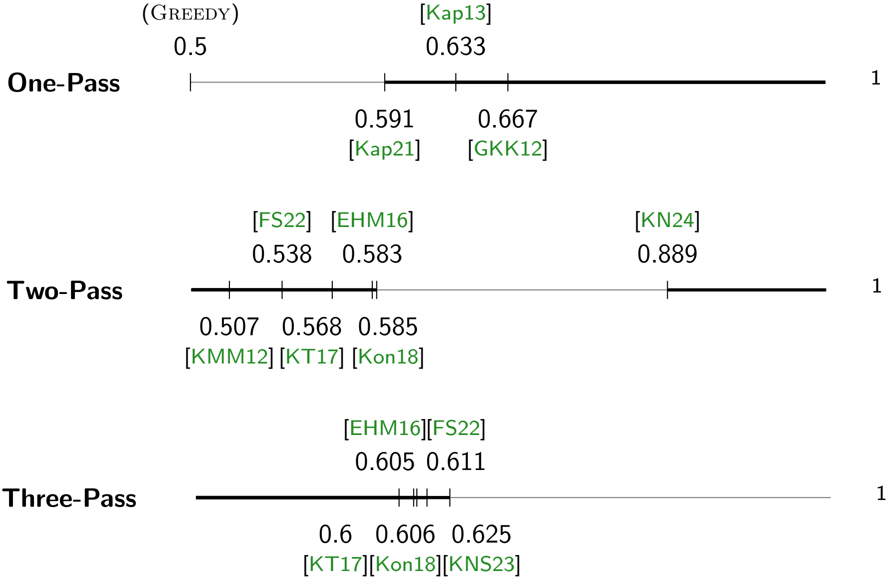
Approximate MBM using $O(n \operatorname{polylog} n)$ Space



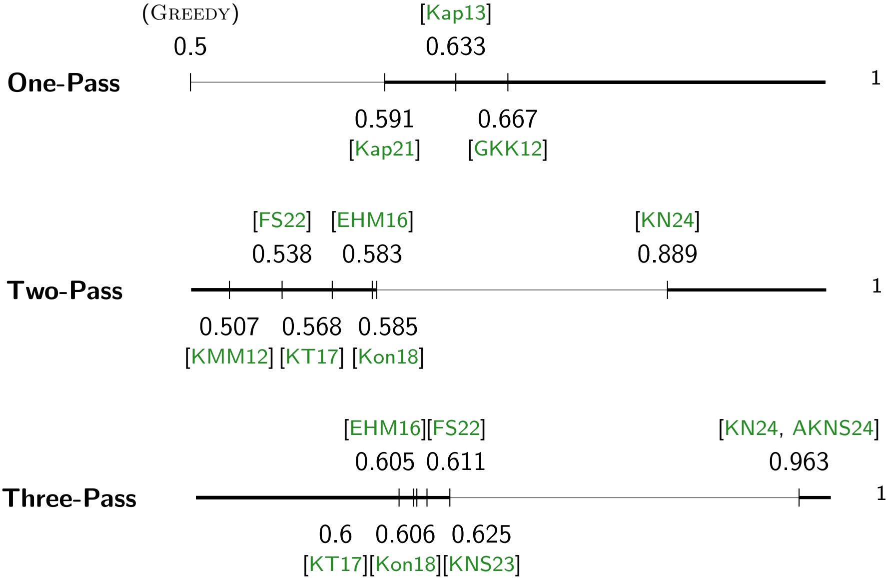
Approximate MBM using $O(n \operatorname{polylog} n)$ Space



Approximate MBM using $O(n \operatorname{polylog} n)$ Space



Approximate MBM using $O(n \operatorname{polylog} n)$ Space



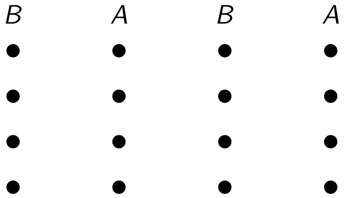
Three-Pass Algorithmic Idea

First Pass

Find a maximal matching M in G using GREEDY.

Subsequent Passes

Find vertex-disjoint augmenting paths (also using GREEDY).



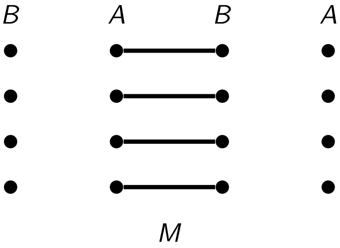
Three-Pass Algorithmic Idea

First Pass

Find a maximal matching M in G using $\textsc{Greedy}$.

Subsequent Passes

Find vertex-disjoint augmenting paths (also using $\textsc{Greedy}$).



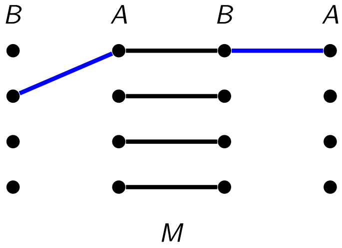
Three-Pass Algorithmic Idea

First Pass

Find a maximal matching M in G using $\textsc{Greedy}$.

Subsequent Passes

Find vertex-disjoint augmenting paths (also using $\textsc{Greedy}$).



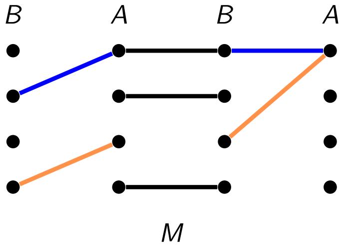
Three-Pass Algorithmic Idea

First Pass

Find a maximal matching M in G using GREEDY.

Subsequent Passes

Find vertex-disjoint augmenting paths (also using GREEDY).



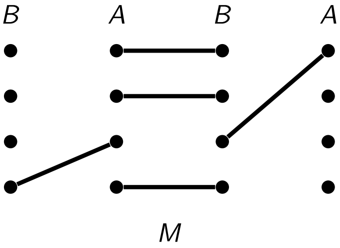
Three-Pass Algorithmic Idea

First Pass

Find a maximal matching M in G using GREEDY.

Subsequent Passes

Find vertex-disjoint augmenting paths (also using GREEDY).



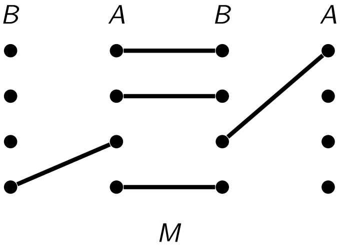
Three-Pass Algorithmic Idea

First Pass

Find a maximal matching M in G using GREEDY.

Subsequent Passes

Find vertex-disjoint augmenting paths (also using GREEDY). **(How?)**



Finding length-3 augmenting paths [KT17]

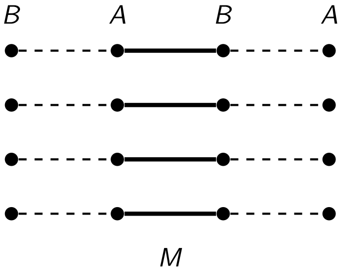
Simple Strategy

1. Find left wings
2. Extend with right wings

Finding length-3 augmenting paths [KT17]

Simple Strategy

1. Find left wings
2. Extend with right wings

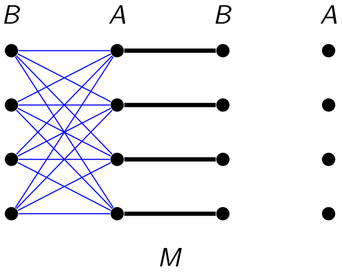


- $|M| = 0.5 \cdot \mu(G)$

Finding length-3 augmenting paths [KT17]

Simple Strategy

1. Find left wings
2. Extend with right wings

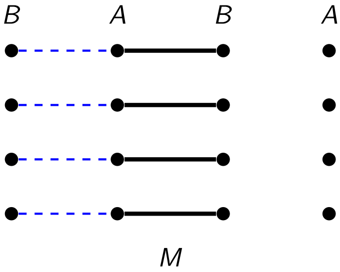


- $|M| = 0.5 \cdot \mu(G)$

Finding length-3 augmenting paths [KT17]

Simple Strategy

1. Find left wings
2. Extend with right wings

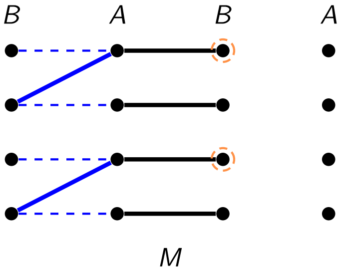


- $|M| = 0.5 \cdot \mu(G)$

Finding length-3 augmenting paths [KT17]

Simple Strategy

1. Find left wings
2. Extend with right wings

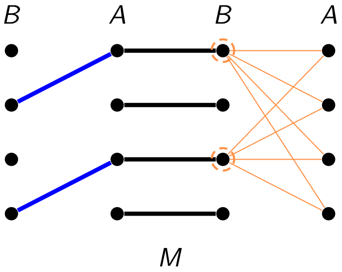


- $|M| = 0.5 \cdot \mu(G)$

Finding length-3 augmenting paths [KT17]

Simple Strategy

1. Find left wings
2. Extend with right wings

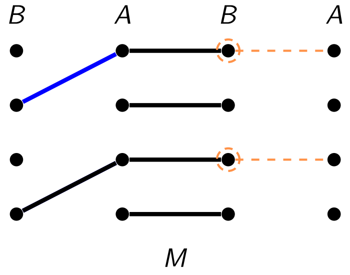


- $|M| = 0.5 \cdot \mu(G)$

Finding length-3 augmenting paths [KT17]

Simple Strategy

1. Find left wings
2. Extend with right wings

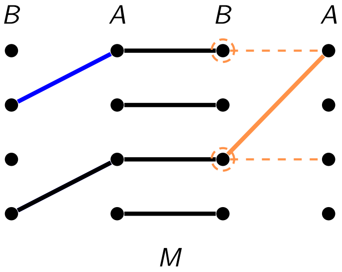


- $|M| = 0.5 \cdot \mu(G)$

Finding length-3 augmenting paths [KT17]

Simple Strategy

1. Find left wings
2. Extend with right wings

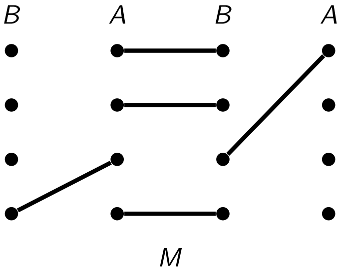


- $|M| = 0.5 \cdot \mu(G)$

Finding length-3 augmenting paths [KT17]

Simple Strategy

1. Find left wings
2. Extend with right wings

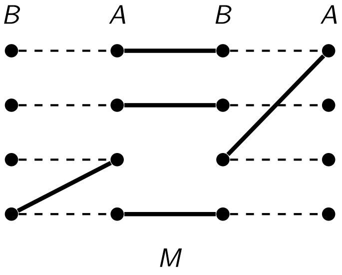


- $|M| = 0.5 \cdot \mu(G)$

Finding length-3 augmenting paths [KT17]

Simple Strategy

1. Find left wings
2. Extend with right wings

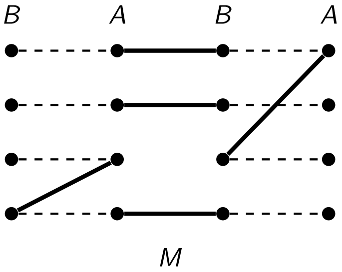


- $|M| = 0.5 \cdot \mu(G)$
- 0.625-approximation
- Not hard!

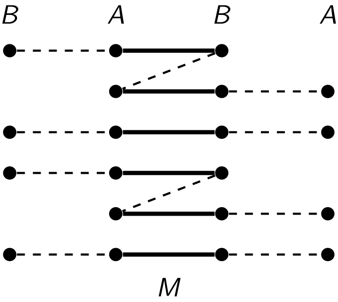
Finding length-3 augmenting paths [KT17]

Simple Strategy

1. Find left wings
2. Extend with right wings



- $|M| = 0.5 \cdot \mu(G)$
- 0.625-approximation
- Not hard!

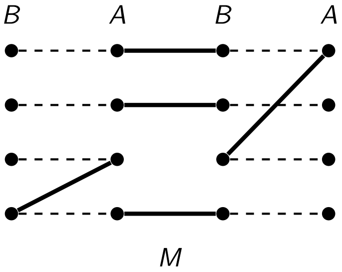


- $|M| = 0.6 \cdot \mu(G)$

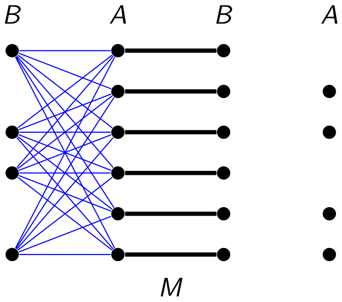
Finding length-3 augmenting paths [KT17]

Simple Strategy

1. Find left wings
2. Extend with right wings



- $|M| = 0.5 \cdot \mu(G)$
- 0.625-approximation
- Not hard!

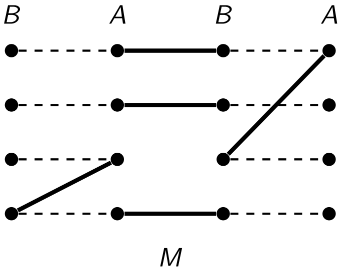


- $|M| = 0.6 \cdot \mu(G)$

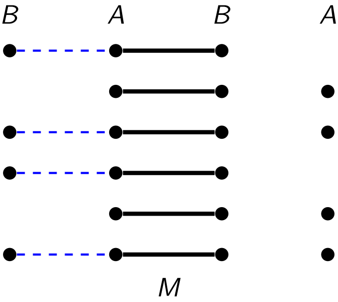
Finding length-3 augmenting paths [KT17]

Simple Strategy

1. Find left wings
2. Extend with right wings



- $|M| = 0.5 \cdot \mu(G)$
- 0.625-approximation
- Not hard!

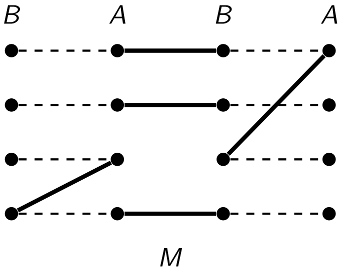


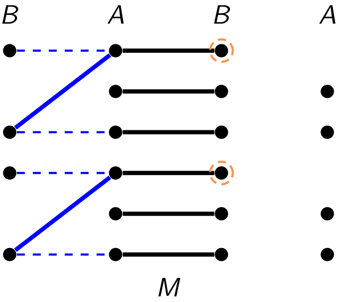
- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 augmenting paths [KT17]

Simple Strategy

1. Find left wings
2. Extend with right wings





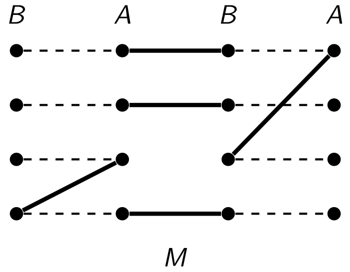
- $|M| = 0.5 \cdot \mu(G)$
- 0.625-approximation
- Not hard!

- $|M| = 0.6 \cdot \mu(G)$

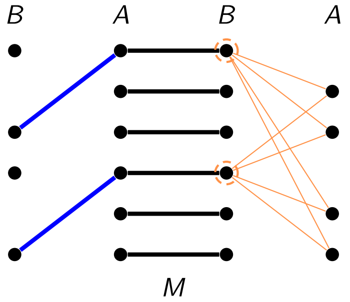
Finding length-3 augmenting paths [KT17]

Simple Strategy

1. Find left wings
2. Extend with right wings



- $|M| = 0.5 \cdot \mu(G)$
- 0.625-approximation
- Not hard!

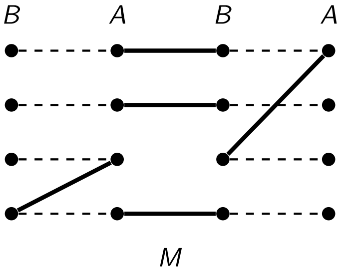


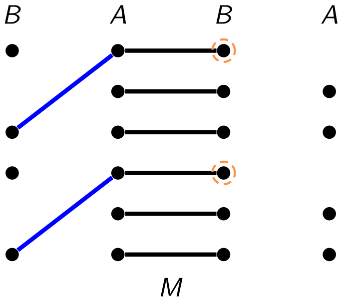
- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 augmenting paths [KT17]

Simple Strategy

1. Find left wings
2. Extend with right wings





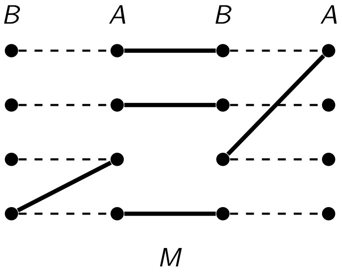
- $|M| = 0.5 \cdot \mu(G)$
- 0.625-approximation
- Not hard!

- $|M| = 0.6 \cdot \mu(G)$

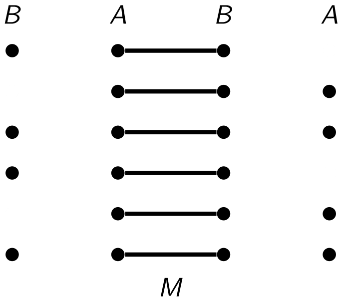
Finding length-3 augmenting paths [KT17]

Simple Strategy

1. Find left wings
2. Extend with right wings



- $|M| = 0.5 \cdot \mu(G)$
- 0.625-approximation
- Not hard!

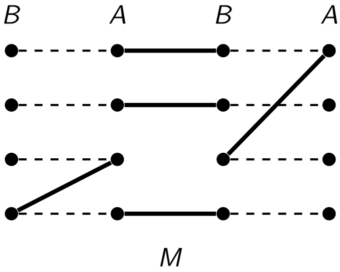


- $|M| = 0.6 \cdot \mu(G)$

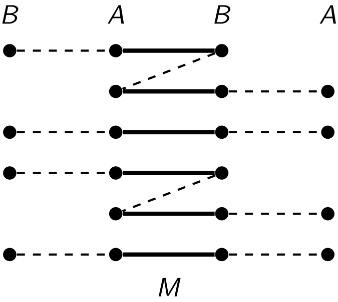
Finding length-3 augmenting paths [KT17]

Simple Strategy

1. Find left wings
2. Extend with right wings



- $|M| = 0.5 \cdot \mu(G)$
- 0.625-approximation
- Not hard!



- $|M| = 0.6 \cdot \mu(G)$
- 0.6-approximation
- **Hard instance!**

Finding length-3 & length-5 augmenting paths [KNS23]

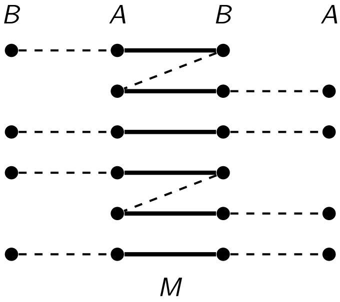
Our Strategy

1. Find left and right wings
2. Extend paths to either length-3 or length-5 augmenting paths

Finding length-3 & length-5 augmenting paths [KNS23]

Our Strategy

1. Find left and right wings
2. Extend paths to either length-3 or length-5 augmenting paths

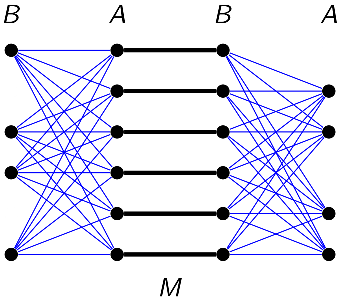


- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 & length-5 augmenting paths [KNS23]

Our Strategy

1. Find left and right wings
2. Extend paths to either length-3 or length-5 augmenting paths

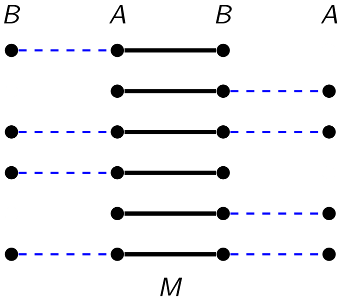


- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 & length-5 augmenting paths [KNS23]

Our Strategy

1. Find left and right wings
2. Extend paths to either length-3 or length-5 augmenting paths

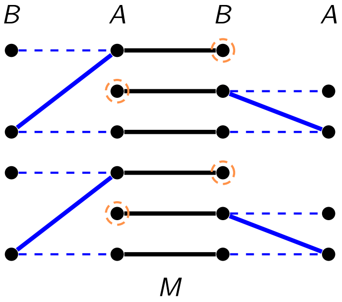


- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 & length-5 augmenting paths [KNS23]

Our Strategy

1. Find left and right wings
2. Extend paths to either length-3 or length-5 augmenting paths

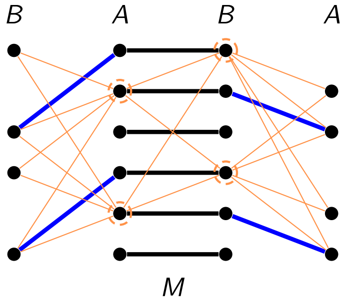


- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 & length-5 augmenting paths [KNS23]

Our Strategy

1. Find left and right wings
2. Extend paths to either length-3 or length-5 augmenting paths

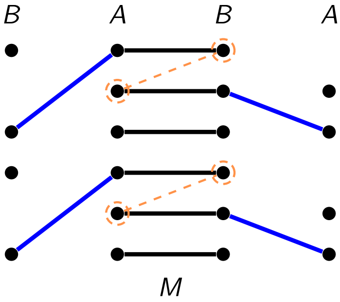


- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 & length-5 augmenting paths [KNS23]

Our Strategy

1. Find left and right wings
2. Extend paths to either length-3 or length-5 augmenting paths

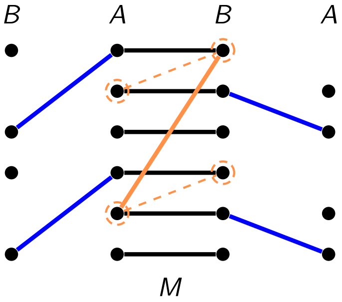


- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 & length-5 augmenting paths [KNS23]

Our Strategy

1. Find left and right wings
2. Extend paths to either length-3 or length-5 augmenting paths

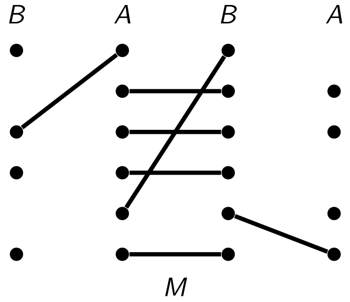


- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 & length-5 augmenting paths [KNS23]

Our Strategy

1. Find left and right wings
2. Extend paths to either length-3 or length-5 augmenting paths

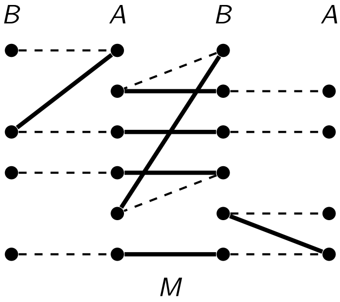


- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 & length-5 augmenting paths [KNS23]

Our Strategy

1. Find left and right wings
2. Extend paths to either length-3 or length-5 augmenting paths

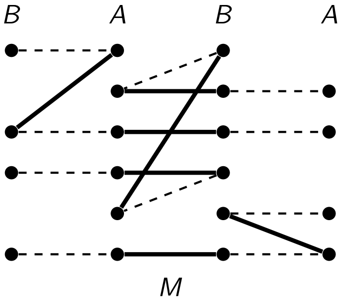


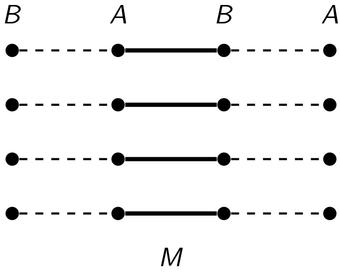
- $|M| = 0.6 \cdot \mu(G)$
- 0.7-approximation
- Not hard anymore!

Finding length-3 & length-5 augmenting paths [KNS23]

Our Strategy

1. Find left and right wings
2. Extend paths to either length-3 or length-5 augmenting paths





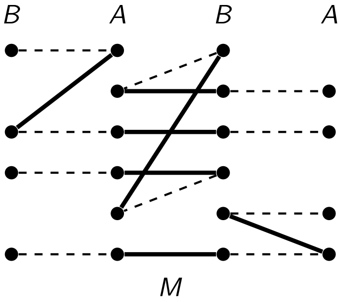
- $|M| = 0.6 \cdot \mu(G)$
- 0.7-approximation
- Not hard anymore!

- $|M| = 0.5 \cdot \mu(G)$

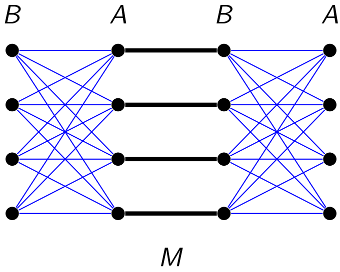
Finding length-3 & length-5 augmenting paths [KNS23]

Our Strategy

1. Find left and right wings
2. Extend paths to either length-3 or length-5 augmenting paths



- $|M| = 0.6 \cdot \mu(G)$
- 0.7-approximation
- Not hard anymore!

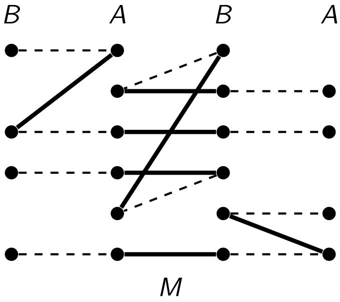


- $|M| = 0.5 \cdot \mu(G)$

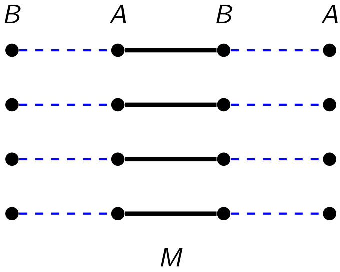
Finding length-3 & length-5 augmenting paths [KNS23]

Our Strategy

1. Find left and right wings
2. Extend paths to either length-3 or length-5 augmenting paths



- $|M| = 0.6 \cdot \mu(G)$
- 0.7-approximation
- Not hard anymore!

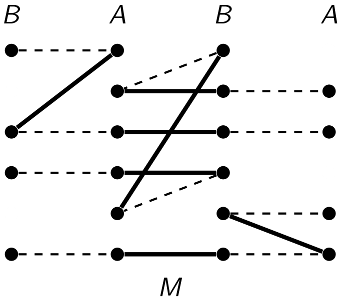


- $|M| = 0.5 \cdot \mu(G)$

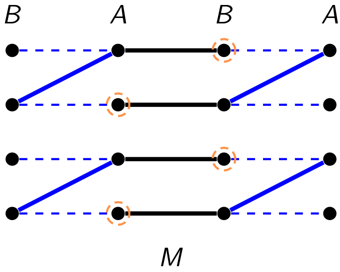
Finding length-3 & length-5 augmenting paths [KNS23]

Our Strategy

1. Find left and right wings
2. Extend paths to either length-3 or length-5 augmenting paths



- $|M| = 0.6 \cdot \mu(G)$
- 0.7-approximation
- Not hard anymore!

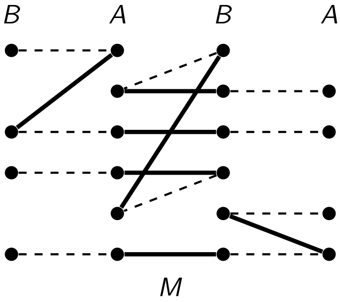


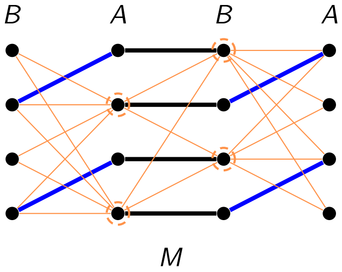
- $|M| = 0.5 \cdot \mu(G)$

Finding length-3 & length-5 augmenting paths [KNS23]

Our Strategy

1. Find left and right wings
2. Extend paths to either length-3 or length-5 augmenting paths





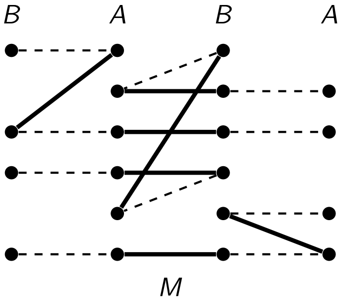
- $|M| = 0.6 \cdot \mu(G)$
- 0.7-approximation
- Not hard anymore!

- $|M| = 0.5 \cdot \mu(G)$

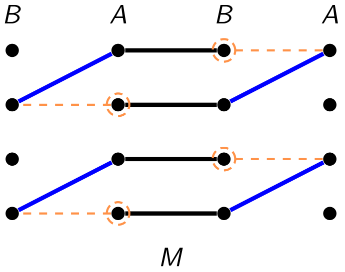
Finding length-3 & length-5 augmenting paths [KNS23]

Our Strategy

1. Find left and right wings
2. Extend paths to either length-3 or length-5 augmenting paths



- $|M| = 0.6 \cdot \mu(G)$
- 0.7-approximation
- Not hard anymore!

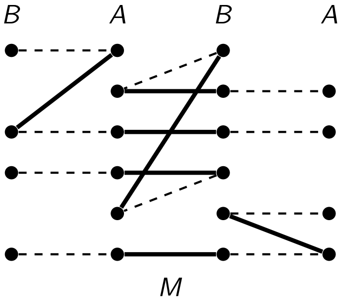


- $|M| = 0.5 \cdot \mu(G)$

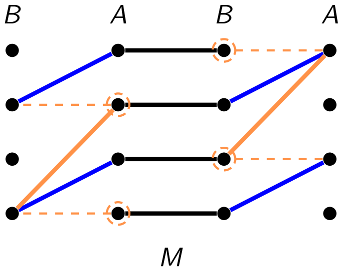
Finding length-3 & length-5 augmenting paths [KNS23]

Our Strategy

1. Find left and right wings
2. Extend paths to either length-3 or length-5 augmenting paths



- $|M| = 0.6 \cdot \mu(G)$
- 0.7-approximation
- Not hard anymore!

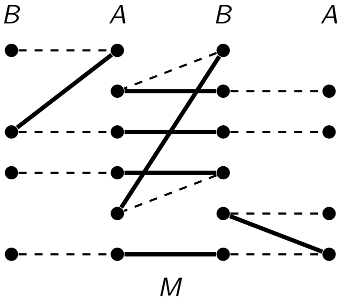


- $|M| = 0.5 \cdot \mu(G)$

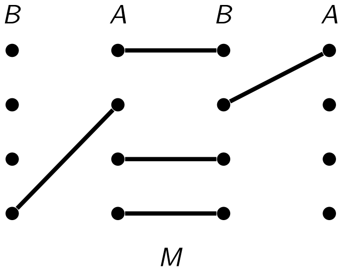
Finding length-3 & length-5 augmenting paths [KNS23]

Our Strategy

1. Find left and right wings
2. Extend paths to either length-3 or length-5 augmenting paths



- $|M| = 0.6 \cdot \mu(G)$
- 0.7-approximation
- Not hard anymore!

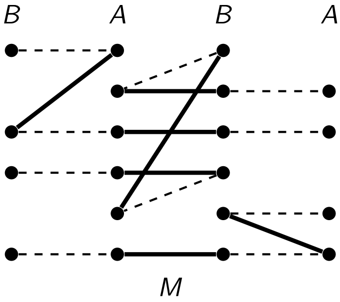


- $|M| = 0.5 \cdot \mu(G)$

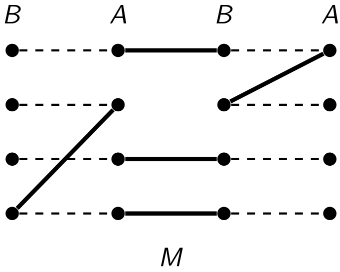
Finding length-3 & length-5 augmenting paths [KNS23]

Our Strategy

1. Find left and right wings
2. Extend paths to either length-3 or length-5 augmenting paths



- $|M| = 0.6 \cdot \mu(G)$
- 0.7-approximation
- Not hard anymore!



- $|M| = 0.5 \cdot \mu(G)$
- 0.625-approximation
- **Hard instance!**

Our Analysis [KNS23]

Main Lemma

Let $|M| = (0.5 + \epsilon) \cdot \mu(G)$ for $\epsilon \geq 0$, then our strategy finds

$$(0.125 - \frac{3}{4}\epsilon) \cdot \mu(G)$$

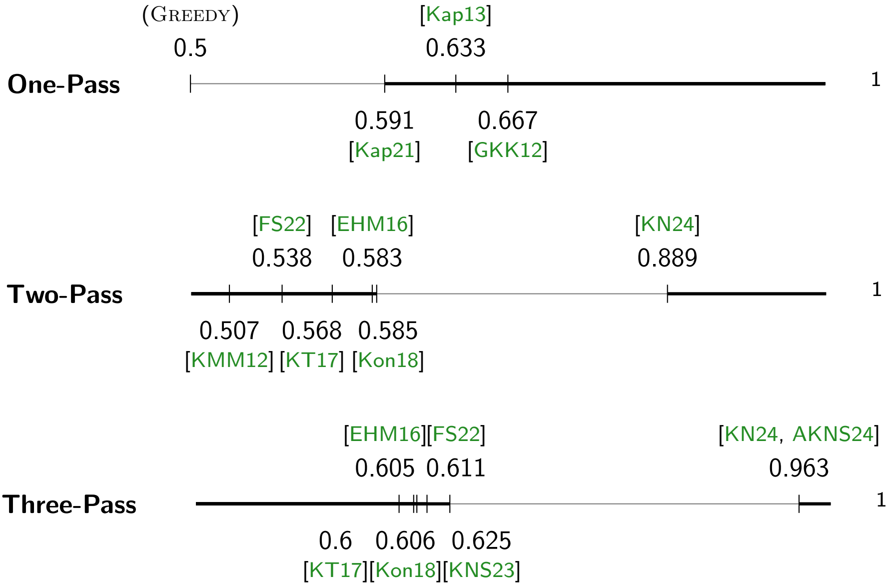
vertex-disjoint augmenting paths and the large matching found is of size

$$(0.625 + \frac{\epsilon}{4}) \cdot \mu(G).$$

Space Used

$\textsc{Greedy}$ only stores $O(n)$ edges in each pass $\implies$ $O(n \log n)$ space.

Conclusion


One-Pass
(GREEDY)
0.5
[Kap13]
0.633
0.591
[Kap21]
0.667
[GKK12]
1
Two-Pass
[FS22]
0.538
[EHM16]
0.583
[KN24]
0.889
0.507
[KMM12]
0.568
[KT17]
0.585
[Kon18]
1
Three-Pass
[EHM16][FS22]
0.605 0.611
[KN24, AKNS24]
0.963
0.6
[KT17]
0.606
[Kon18]
0.625
[KNS23]
1

References I

Sepehr Assadi, Christian Konrad, Kheeran K. Naidu, and Janani Sundaresan, *O(log log n) passes is optimal for semi-streaming maximal independent set*, Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, British Columbia, Canada, June 24-28, 2024, ACM, 2024.

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.

References II

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.

Moran Feldman and Ariel Szarf, *Maximum matching sans maximal matching: A new approach for finding maximum matchings in the data stream model*, Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2022, September 19-21, 2022, University of Illinois, Urbana-Champaign, USA (Virtual Conference) (Amit Chakrabarti and Chaitanya Swamy, eds.), LIPIcs, vol. 245, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, pp. 33:1–33:24.

References III

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.

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.

References IV

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, *An unconditional lower bound for two-pass streaming algorithms for maximum matching approximation*, Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, Virginia, USA, January 7-10, SIAM, 2024, pp. 2881–2899.

References V

Christian Konrad, Kheeran K. Naidu, and Arun Steward, *Maximum matching via maximal matching queries*, 40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023, March 7-9, 2023, Hamburg, Germany (Petra Berenbrink, Patricia Bouyer, Anuj Dawar, and Mamadou Moustapha Kanté, eds.), LIPIcs, vol. 254, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, pp. 41:1–41:22.

Christian Konrad, *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.

References VI

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.