Maximum Matching via Maximal Matching Queries

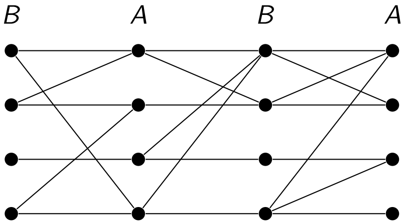
Christian Konrad, **Kheeran K. Naidu** & Arun Steward

University of Bristol, UK

{*christian.konrad,kheeran.naidu,cw18903*}*@bristol.ac.uk*

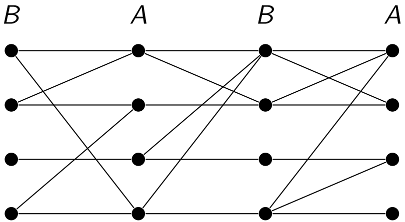
Preliminaries

In this talk, we consider bipartite graphs $G = (A, B, E)$.



Preliminaries

In this talk, we consider bipartite graphs $G = (A, B, E)$.

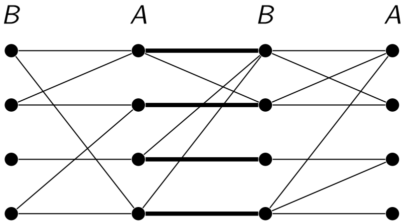


Matchings

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

Preliminaries

In this talk, we consider bipartite graphs $G = (A, B, E)$.



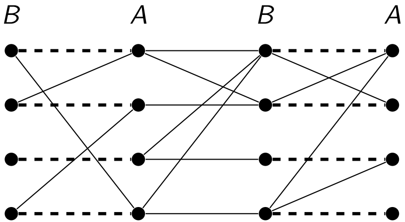
Matchings

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

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

Preliminaries

In this talk, we consider bipartite graphs $G = (A, B, E)$.



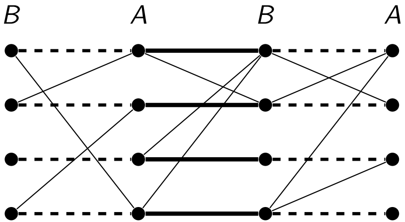
Matchings

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

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

Preliminaries

In this talk, we consider bipartite graphs $G = (A, B, E)$.



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.

Edge Query Model

Algorithm's Goal

Return a **large matching** of the bipartite input graph $G = (A, B, E)$ using only **deterministic edge queries** to a **maximal matching oracle**.

Edge Query Model

Algorithm's Goal

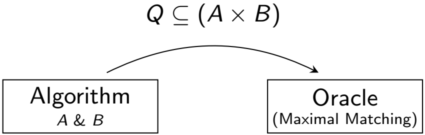
Return a **large matching** of the bipartite input graph $G = (A, B, E)$ using only **deterministic edge queries** to a **maximal matching oracle**.

Algorithm
A & B

Edge Query Model

Algorithm's Goal

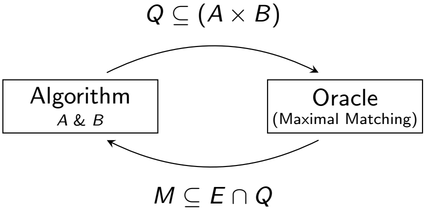
Return a **large matching** of the bipartite input graph $G = (A, B, E)$ using only **deterministic edge queries** to a **maximal matching oracle**.



Edge Query Model

Algorithm's Goal

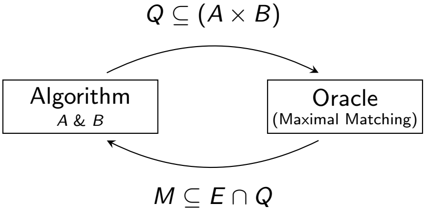
Return a **large matching** of the bipartite input graph $G = (A, B, E)$ using only **deterministic edge queries** to a **maximal matching oracle**.



Edge Query Model

Algorithm's Goal

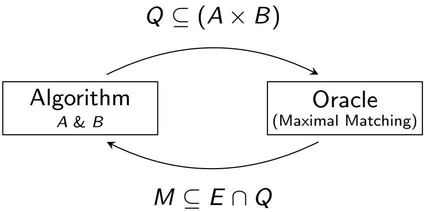
Return a **large matching** of the bipartite input graph $G = (A, B, E)$ using only **deterministic edge queries** to a **maximal matching oracle**.



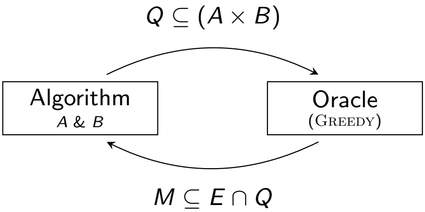
Motivation

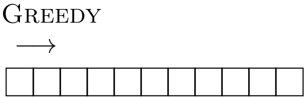
Computing a maximal matching is **easy** in various computational models such as **data streaming** and Massively Parallel Computation.

Edge Query Model and Data Streaming

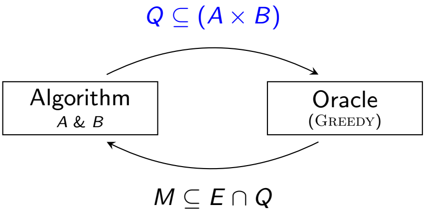


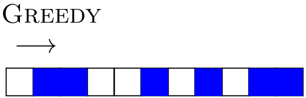
Edge Query Model and Data Streaming



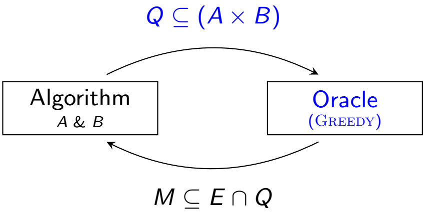


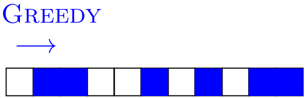
Edge Query Model and Data Streaming



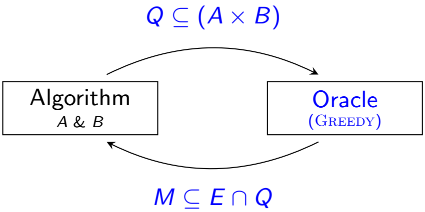


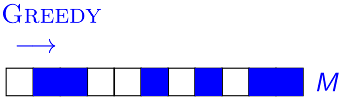
Edge Query Model and Data Streaming



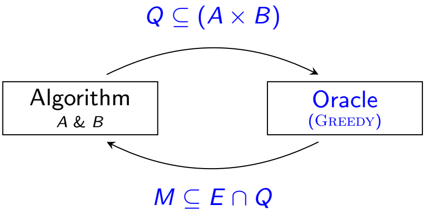


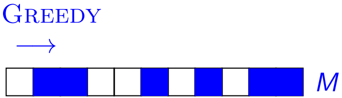
Edge Query Model and Data Streaming





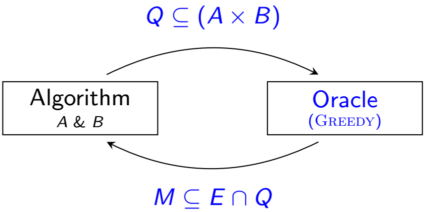
Edge Query Model and Data Streaming

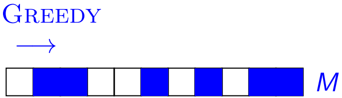




As long as Q can be **specified in $\tilde{O}(n)$ space**, each round can be implemented in **one pass** of the stream using **semi-streaming space**.

Edge Query Model and Data Streaming





As long as Q can be **specified in $\tilde{O}(n)$ space**, each round can be implemented in **one pass** of the stream using **semi-streaming space**.

Known Algorithms

1. 0.6-approximation **MBM** in 3-passes [KT17] (see also [KMM12, FKM+05]) – state-of-the-art is 0.611-approximation [FS22].
2. $(1-\epsilon)$-approximation **MBM** in $O(\frac{1}{\epsilon^2})$-passes [ALT21] – current state-of-the-art.

Our Results

Our Results

Algorithm

0.625-approximation algorithm in **3-rounds** of the deterministic edge-query model.

Implies a **3-pass semi-streaming** algorithm for **MBM** (state-of-the-art – improving on 0.611 [FS22]).

Our Results

Algorithm

0.625-approximation algorithm in **3-rounds** of the deterministic edge-query model.

Implies a **3-pass semi-streaming** algorithm for **MBM** (state-of-the-art – improving on 0.611 [FS22]).

Lower Bounds

There **does not exist** a deterministic algorithm for **MM** (even for **MBM**) in the edge query model that achieves a better than

1. 0.5-approximation in **1 round**,
2. $(0.5 + o(1))$-approximation in **2 rounds**, and
3. $(0.625 + o(1))$-approximation in **3 rounds**.

Our Results

Algorithm

0.625-approximation algorithm in **3-rounds** of the deterministic edge-query model.

Implies a **3-pass semi-streaming** algorithm for **MBM** (state-of-the-art – improving on 0.611 [FS22]).

Lower Bounds

There **does not exist** a deterministic algorithm for **MM** (even for **MBM**) in the edge query model that achieves a better than

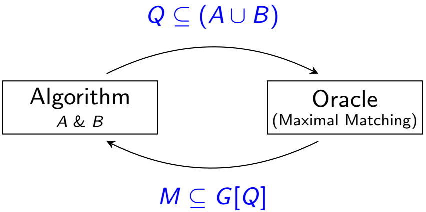
1. 0.5-approximation in **1 round**,
2. $(0.5 + o(1))$-approximation in **2 rounds**, and
3. $(0.625 + o(1))$-approximation in **3 rounds**.

Algorithm is optimal!

Previous Related Work

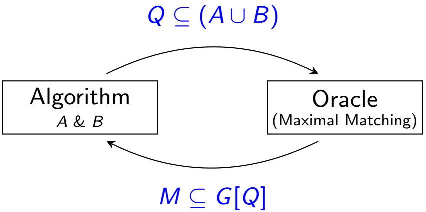
Previous Related Work

Vertex Query Model ([bKK20])



Previous Related Work

Vertex Query Model ([bKK20])



# Rounds	Vertex Query	Edge Query
1	0.5	0.5
2	0.5	$0.5 + o(1)$
3	0.6	$0.625 + o(1)$

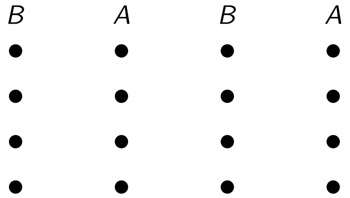
Algorithmic Idea

First Round

Find a maximal matching M in G by querying the complete graph $Q = A \times B$.

Subsequent Rounds

Find vertex-disjoint augmenting paths.



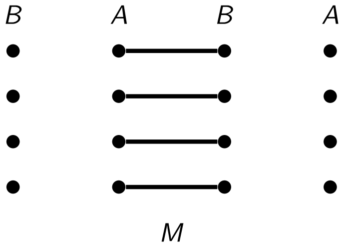
Algorithmic Idea

First Round

Find a maximal matching M in G by querying the complete graph $Q = A \times B$.

Subsequent Rounds

Find vertex-disjoint augmenting paths.



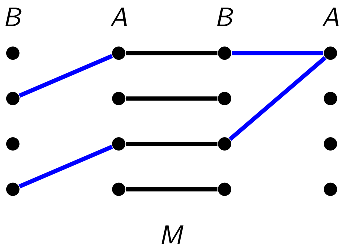
Algorithmic Idea

First Round

Find a maximal matching M in G by querying the complete graph $Q = A \times B$.

Subsequent Rounds

Find vertex-disjoint augmenting paths.



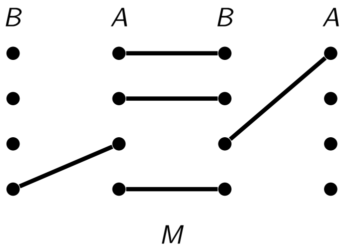
Algorithmic Idea

First Round

Find a maximal matching M in G by querying the complete graph $Q = A \times B$.

Subsequent Rounds

Find vertex-disjoint augmenting paths.



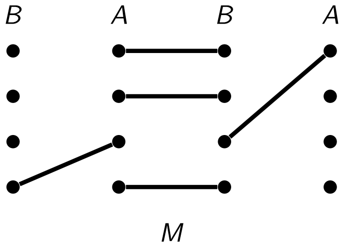
Algorithmic Idea

First Round

Find a maximal matching M in G by querying the complete graph $Q = A \times B$.

Subsequent Rounds

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



Finding length-3 augmenting paths

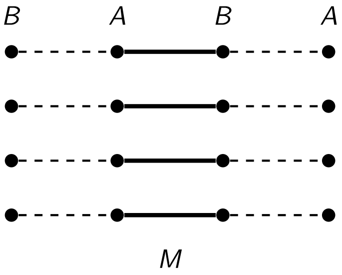
Simple Strategy

1. Find left wings
2. Extend with right wings

Finding length-3 augmenting paths

Simple Strategy

1. Find left wings
2. Extend with right wings

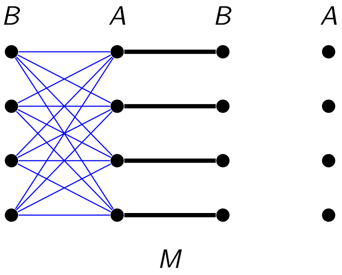


- $|M| = 0.5 \cdot \mu(G)$

Finding length-3 augmenting paths

Simple Strategy

1. Find left wings
2. Extend with right wings

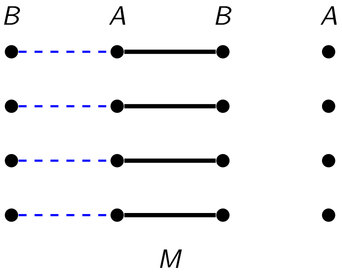


- $|M| = 0.5 \cdot \mu(G)$

Finding length-3 augmenting paths

Simple Strategy

1. Find left wings
2. Extend with right wings

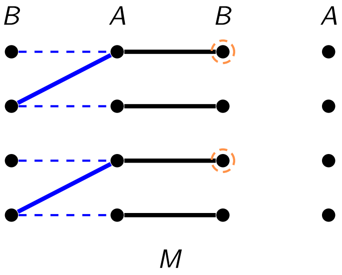


- $|M| = 0.5 \cdot \mu(G)$

Finding length-3 augmenting paths

Simple Strategy

1. Find left wings
2. Extend with right wings

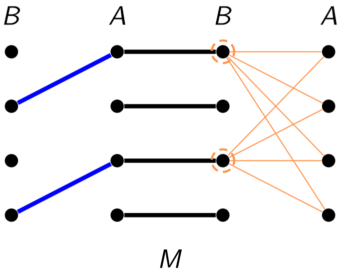


- $|M| = 0.5 \cdot \mu(G)$

Finding length-3 augmenting paths

Simple Strategy

1. Find left wings
2. Extend with right wings

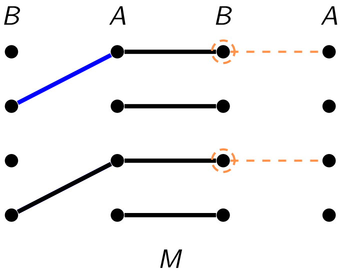


- $|M| = 0.5 \cdot \mu(G)$

Finding length-3 augmenting paths

Simple Strategy

1. Find left wings
2. Extend with right wings

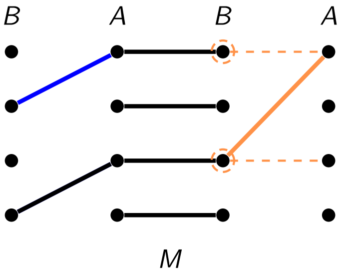


- $|M| = 0.5 \cdot \mu(G)$

Finding length-3 augmenting paths

Simple Strategy

1. Find left wings
2. Extend with right wings

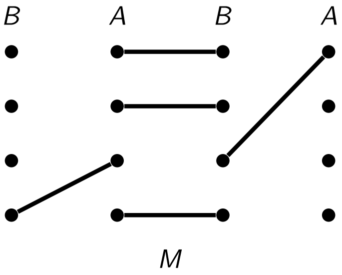


- $|M| = 0.5 \cdot \mu(G)$

Finding length-3 augmenting paths

Simple Strategy

1. Find left wings
2. Extend with right wings

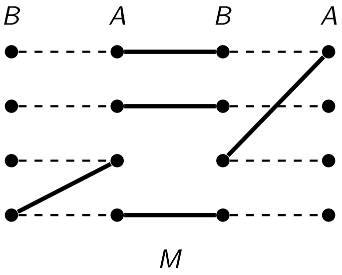


- $|M| = 0.5 \cdot \mu(G)$

Finding length-3 augmenting paths

Simple Strategy

1. Find left wings
2. Extend with right wings

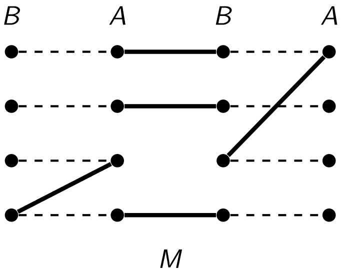


- $|M| = 0.5 \cdot \mu(G)$
- 0.625-approximation
- Not hard!

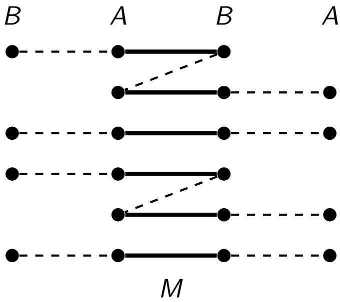
Finding length-3 augmenting paths

Simple Strategy

1. Find left wings
2. Extend with right wings



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

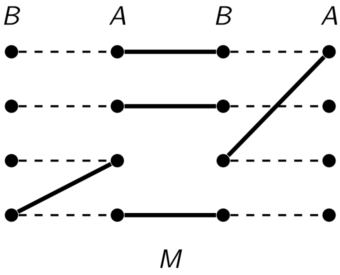


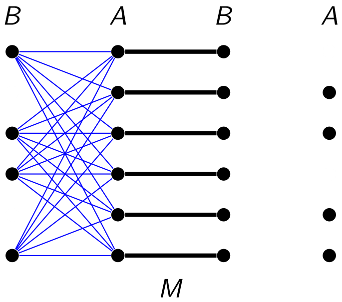
- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 augmenting paths

Simple Strategy

1. Find left wings
2. Extend with right wings





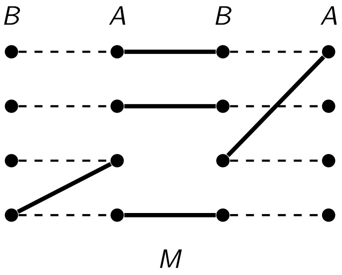
- $|M| = 0.5 \cdot \mu(G)$
- 0.625-approximation
- Not hard!

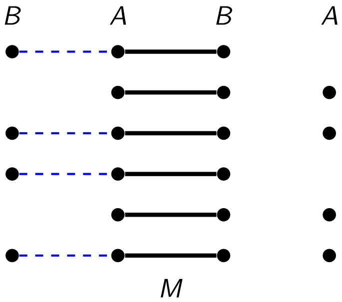
- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 augmenting paths

Simple Strategy

1. Find left wings
2. Extend with right wings





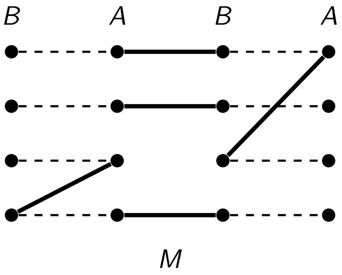
- $|M| = 0.5 \cdot \mu(G)$
- 0.625-approximation
- Not hard!

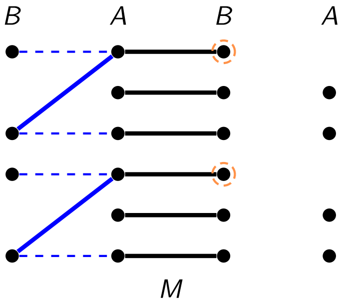
- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 augmenting paths

Simple Strategy

1. Find left wings
2. Extend with right wings





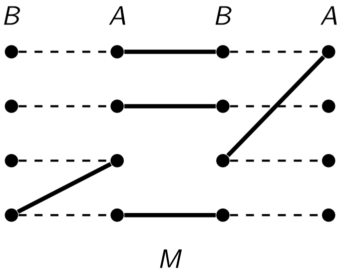
- $|M| = 0.5 \cdot \mu(G)$
- 0.625-approximation
- Not hard!

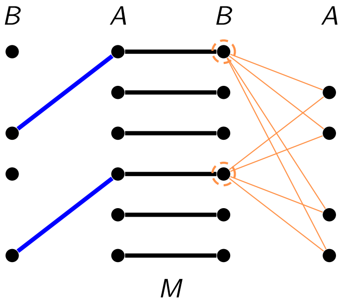
- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 augmenting paths

Simple Strategy

1. Find left wings
2. Extend with right wings





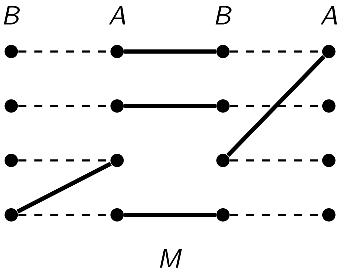
- $|M| = 0.5 \cdot \mu(G)$
- 0.625-approximation
- Not hard!

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

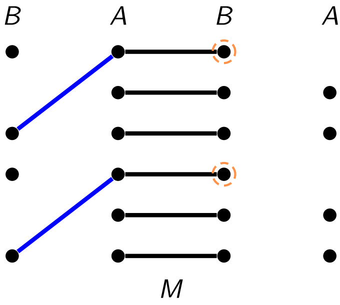
Finding length-3 augmenting paths

Simple Strategy

1. Find left wings
2. Extend with right wings



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

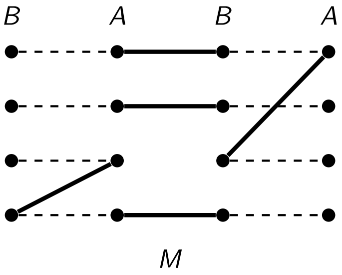


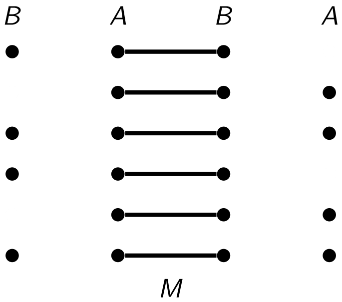
- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 augmenting paths

Simple Strategy

1. Find left wings
2. Extend with right wings





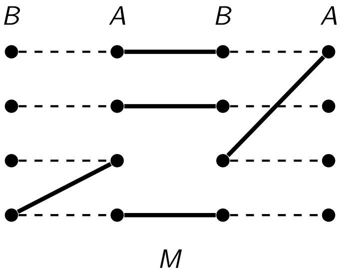
- $|M| = 0.5 \cdot \mu(G)$
- 0.625-approximation
- Not hard!

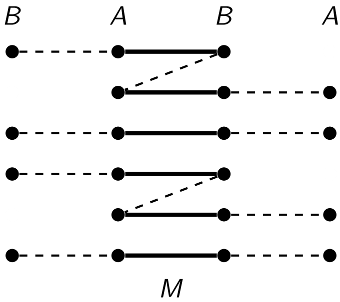
- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 augmenting paths

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

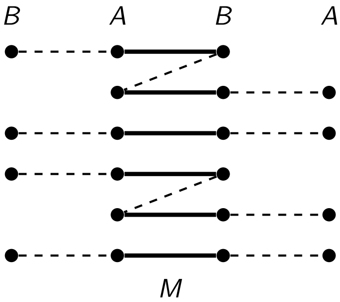
Our Strategy

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

Our Strategy

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

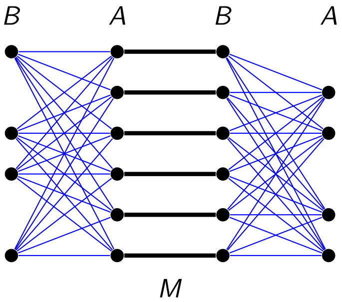


- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 & length-5 augmenting paths

Our Strategy

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

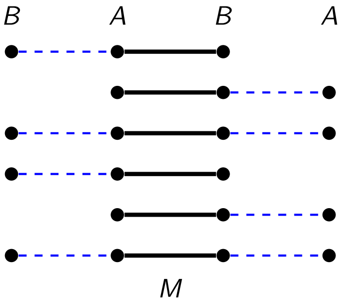


- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 & length-5 augmenting paths

Our Strategy

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

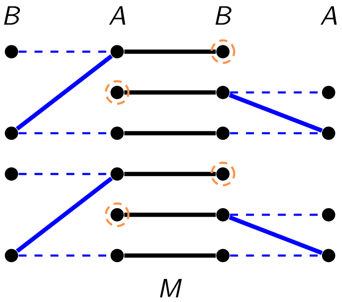


- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 & length-5 augmenting paths

Our Strategy

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

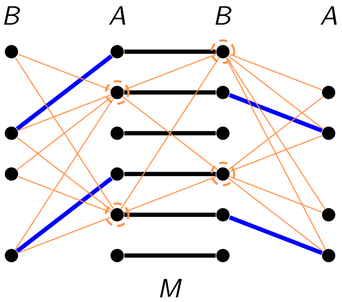


- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 & length-5 augmenting paths

Our Strategy

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

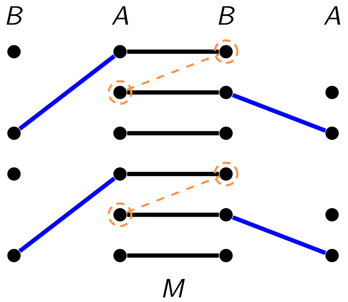


- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 & length-5 augmenting paths

Our Strategy

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

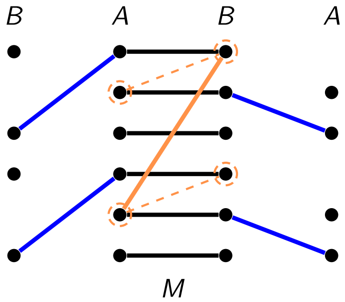


- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 & length-5 augmenting paths

Our Strategy

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

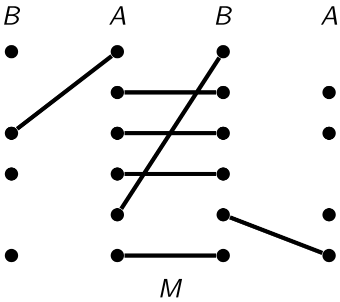


- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 & length-5 augmenting paths

Our Strategy

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

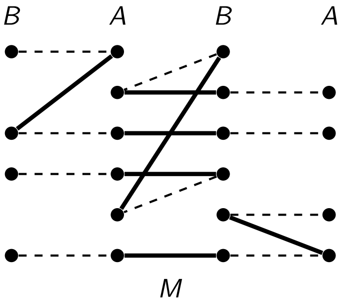


- $|M| = 0.6 \cdot \mu(G)$

Finding length-3 & length-5 augmenting paths

Our Strategy

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

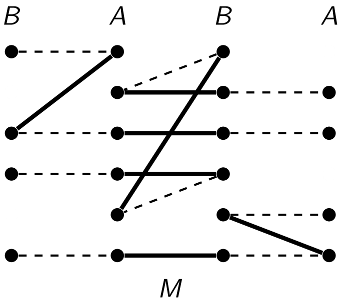


- $|M| = 0.6 \cdot \mu(G)$
- 0.7-approximation
- Not hard anymore!

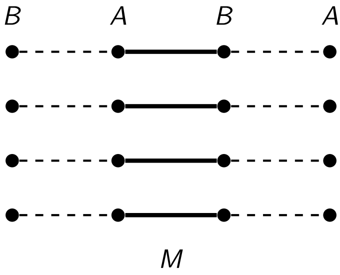
Finding length-3 & length-5 augmenting paths

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!

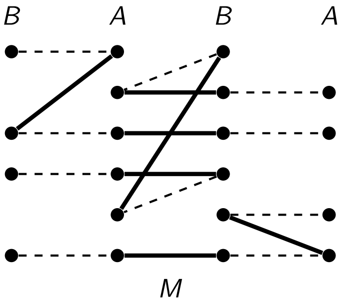


- $|M| = 0.5 \cdot \mu(G)$

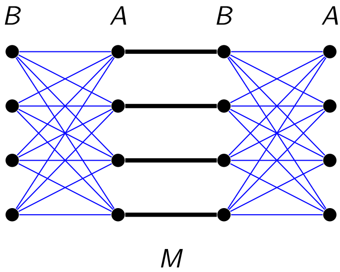
Finding length-3 & length-5 augmenting paths

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!

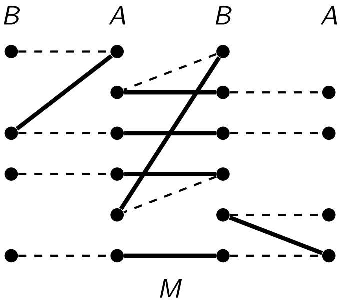


- $|M| = 0.5 \cdot \mu(G)$

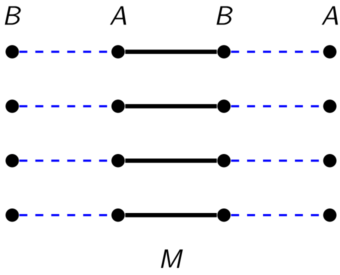
Finding length-3 & length-5 augmenting paths

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!

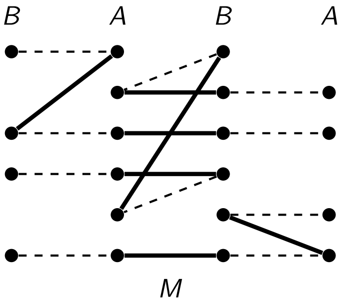


- $|M| = 0.5 \cdot \mu(G)$

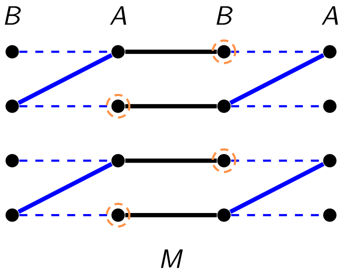
Finding length-3 & length-5 augmenting paths

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!

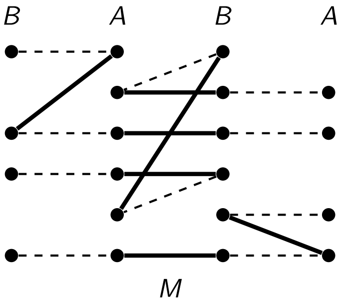


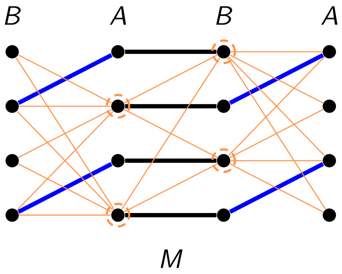
- $|M| = 0.5 \cdot \mu(G)$

Finding length-3 & length-5 augmenting paths

Our Strategy

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





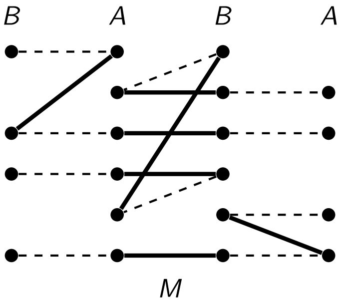
- $|M| = 0.6 \cdot \mu(G)$
- 0.7-approximation
- Not hard anymore!

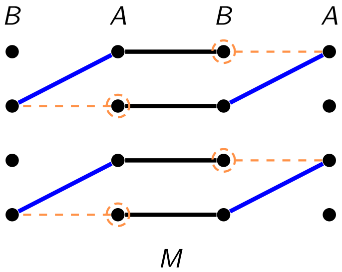
- $|M| = 0.5 \cdot \mu(G)$

Finding length-3 & length-5 augmenting paths

Our Strategy

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





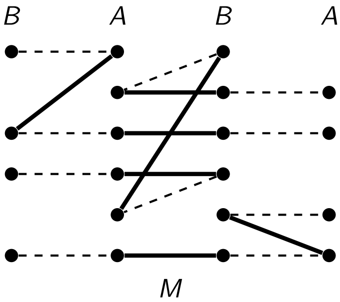
- $|M| = 0.6 \cdot \mu(G)$
- 0.7-approximation
- Not hard anymore!

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

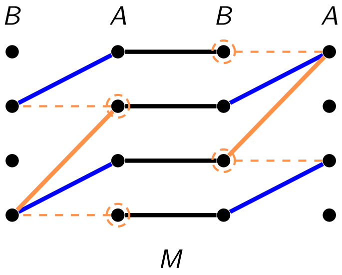
Finding length-3 & length-5 augmenting paths

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!

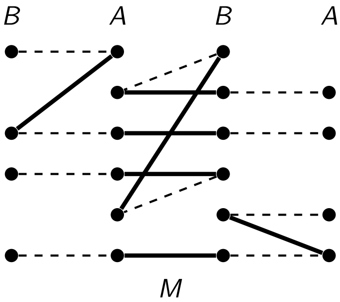


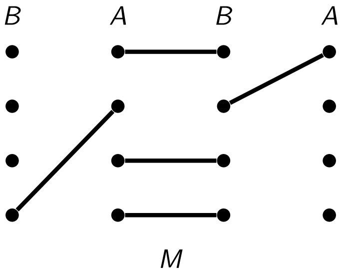
- $|M| = 0.5 \cdot \mu(G)$

Finding length-3 & length-5 augmenting paths

Our Strategy

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





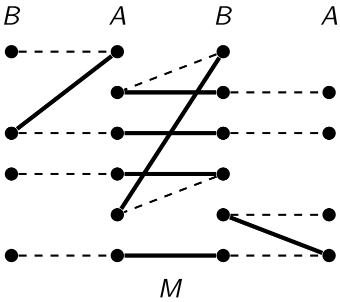
- $|M| = 0.6 \cdot \mu(G)$
- 0.7-approximation
- Not hard anymore!

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

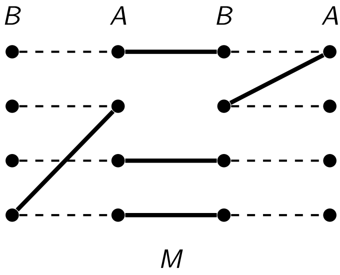
Finding length-3 & length-5 augmenting paths

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

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

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

This is tight for our algorithm.

Our Analysis

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

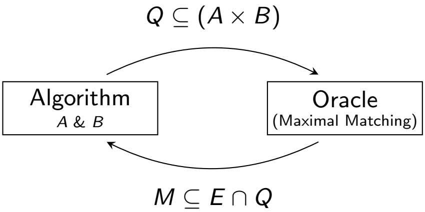
$$(0.625 + \frac{\epsilon}{4}) \cdot \mu(G).$$

This is tight for our algorithm.

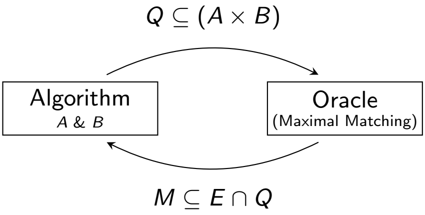
Semi-Streaming

Using GREEDY this immediately gives a 3-pass semi-streaming algorithm with the same guarantees.

Lower Bound Idea



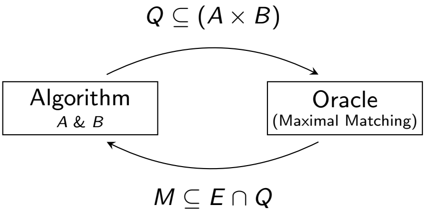
Lower Bound Idea

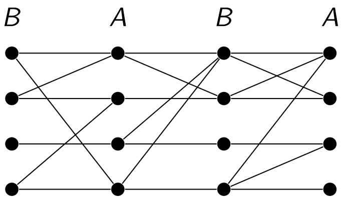


Observation

The algorithm learns about **edges** M and **non-edges** N of G.

Lower Bound Idea

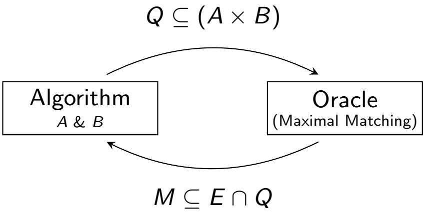


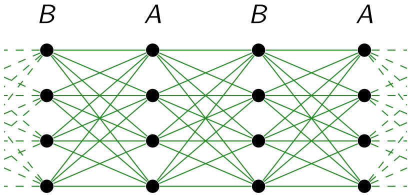


Observation

The algorithm learns about **edges** M and **non-edges** N of G.

Lower Bound Idea

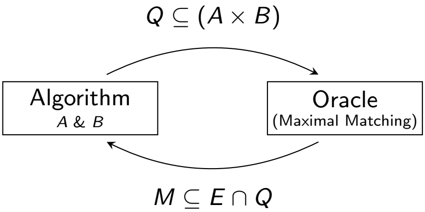


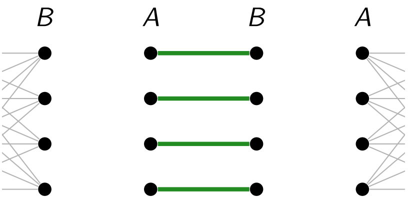


Observation

The algorithm learns about **edges** M and **non-edges** N of G.

Lower Bound Idea

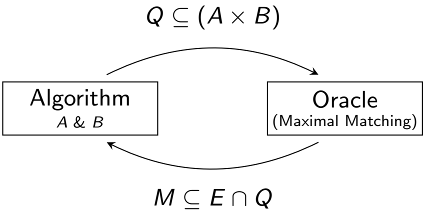


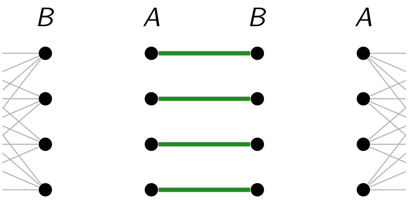


Observation

The algorithm learns about **edges** M and **non-edges** N of G.

Lower Bound Idea



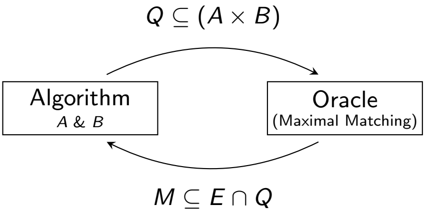


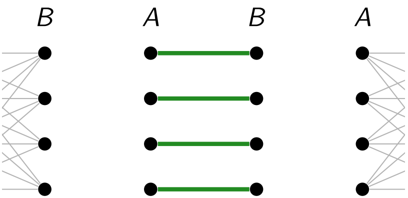
Observation

The algorithm learns about **edges** M and **non-edges** N of G.

Main Idea

Lower Bound Idea





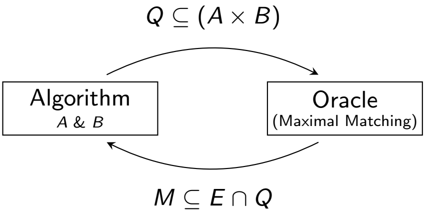
Observation

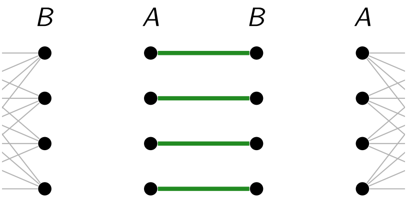
The algorithm learns about **edges** M and **non-edges** N of G.

Main Idea

- Find a hard instance for any sequence of queries $Q_1, Q_2 \ldots$

Lower Bound Idea





Observation

The algorithm learns about **edges** M and **non-edges** N of G.

Main Idea

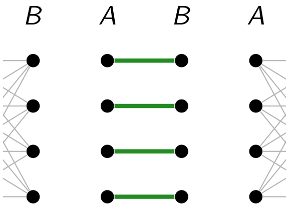
- Find a hard instance for any sequence of queries $Q_1, Q_2 \ldots$
- For any query Q_i, the information committed is a subset of $\tilde{M}_i$ and $\tilde{N}_i$ (up to isomorphisms)

One, Two & Three Query Rounds

One Round **Two Rounds** **Three Rounds**

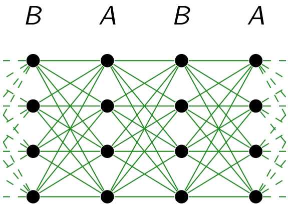
One, Two & Three Query Rounds

One Round | **Two Rounds** | **Three Rounds**



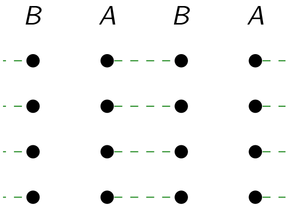
One, Two & Three Query Rounds

One Round **Two Rounds** **Three Rounds**



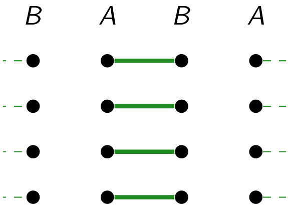
One, Two & Three Query Rounds

One Round | **Two Rounds** | **Three Rounds**



One, Two & Three Query Rounds

One Round

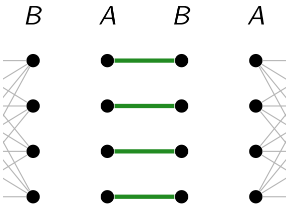


Two Rounds

Three Rounds

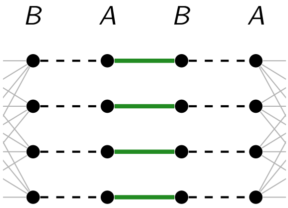
One, Two & Three Query Rounds

<u>One Round</u> **<u>Two Rounds</u>** **<u>Three Rounds</u>**



One, Two & Three Query Rounds

One Round



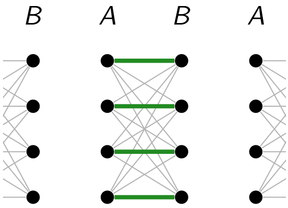
0.5
approx LB

Two Rounds

Three Rounds

One, Two & Three Query Rounds

One Round



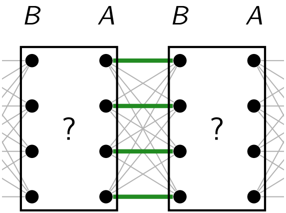
0.5
approx LB

Two Rounds

Three Rounds

One, Two & Three Query Rounds

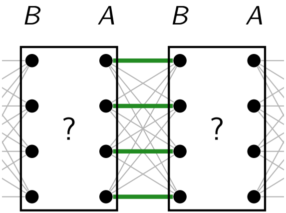
One Round | **Two Rounds** | **Three Rounds**



0.5
approx LB

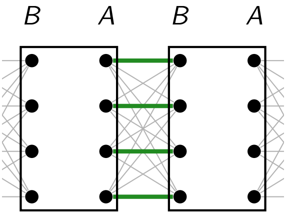
One, Two & Three Query Rounds

One Round



0.5
approx LB

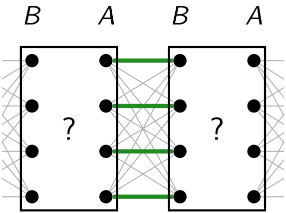
Two Rounds



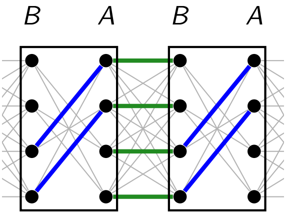
Three Rounds

One, Two & Three Query Rounds

One Round



Two Rounds

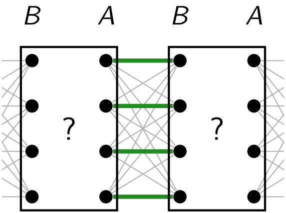


Three Rounds

0.5
approx LB

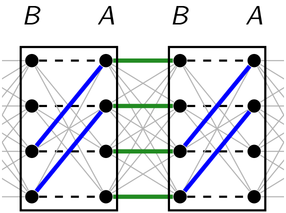
One, Two & Three Query Rounds

One Round



0.5
approx LB

Two Rounds

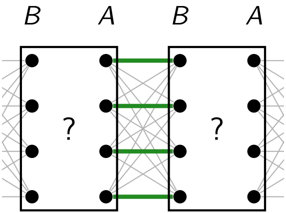


$0.5 + o(1)$
approx LB

Three Rounds

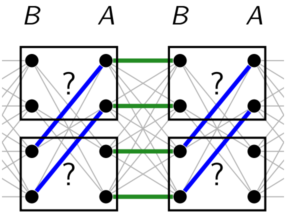
One, Two & Three Query Rounds

One Round



0.5
approx LB

Two Rounds

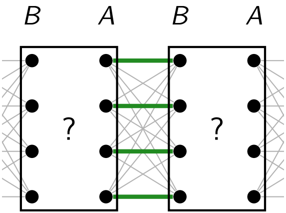


$0.5 + o(1)$
approx LB

Three Rounds

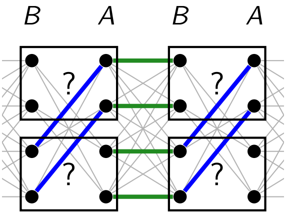
One, Two & Three Query Rounds

One Round



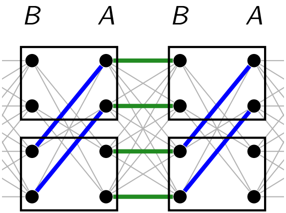
0.5
approx LB

Two Rounds



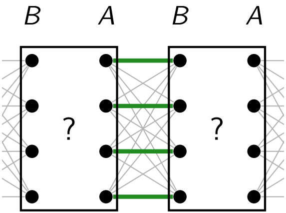
$0.5 + o(1)$
approx LB

Three Rounds



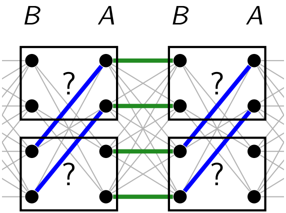
One, Two & Three Query Rounds

One Round



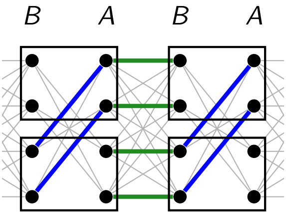
0.5
approx LB

Two Rounds



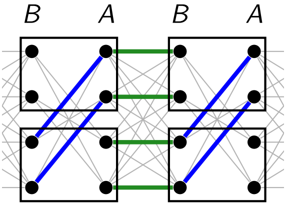
$0.5 + o(1)$
approx LB

Three Rounds



(more involved)

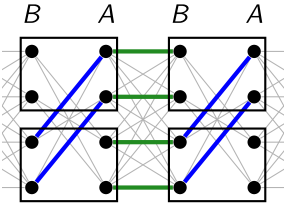
Three Round Proof Sketch I



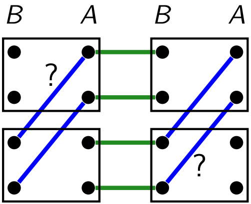
Part A

Part B

Three Round Proof Sketch I

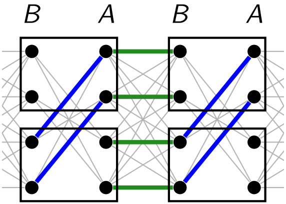


Part A

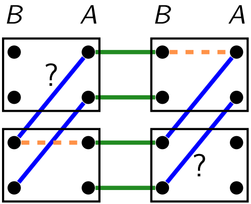


Part B

Three Round Proof Sketch I

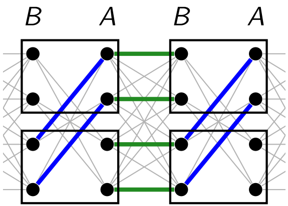


Part A

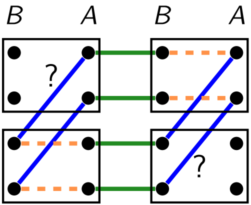


Part B

Three Round Proof Sketch I

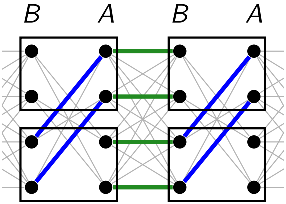


Part A

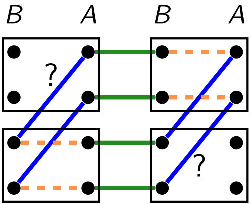


Part B

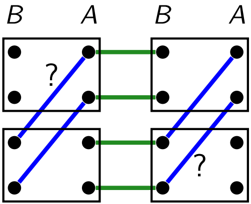
Three Round Proof Sketch I



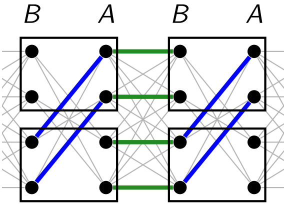
<u>Part A</u>



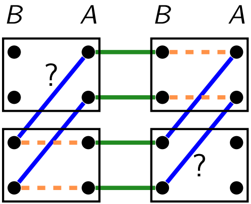
<u>Part B</u>



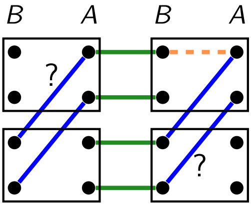
Three Round Proof Sketch I



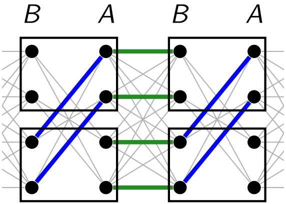
Part A



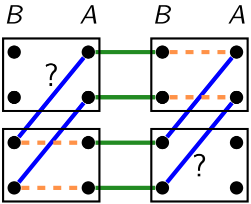
Part B



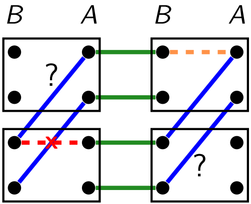
Three Round Proof Sketch I



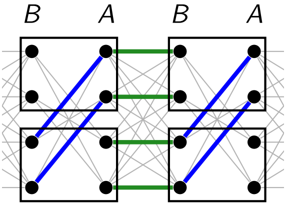
Part A



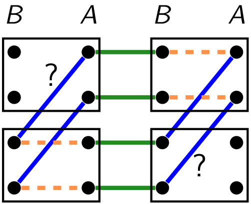
Part B



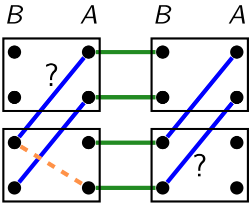
Three Round Proof Sketch I



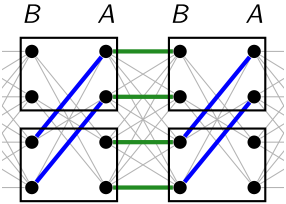
Part A



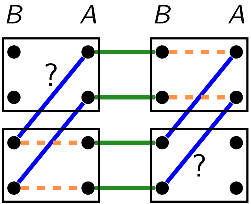
Part B



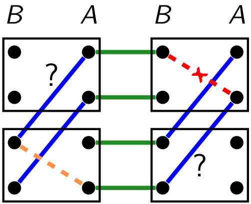
Three Round Proof Sketch I



Part A

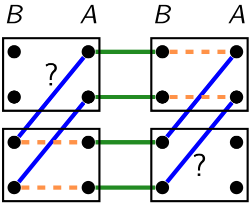


Part B

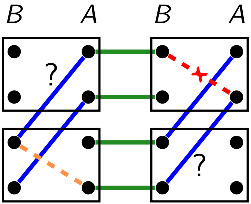


Three Round Proof Sketch II

Part A

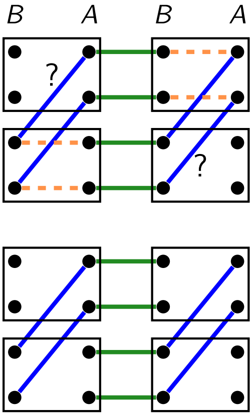


Part B

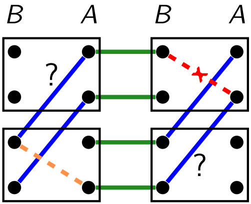


Three Round Proof Sketch II

<u>Part A</u>

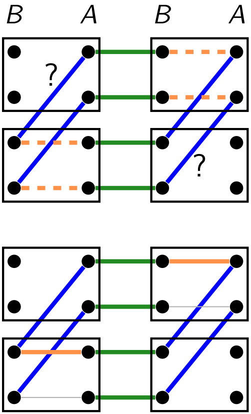


<u>Part B</u>

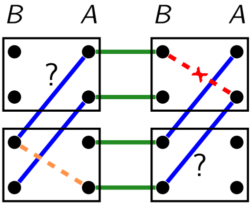


Three Round Proof Sketch II

Part A

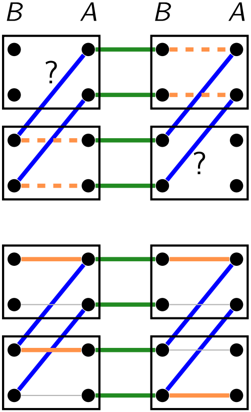


Part B

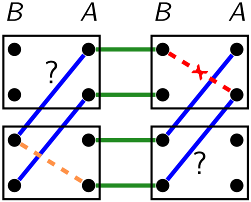


Three Round Proof Sketch II

Part A

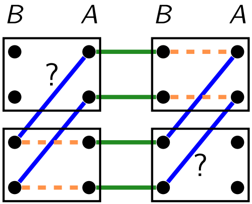


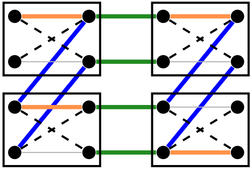
Part B



Three Round Proof Sketch II

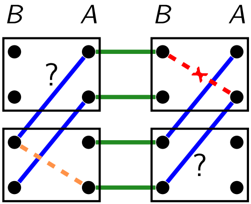
Part A





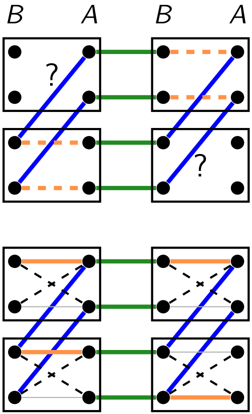
$0.625 + o(1)$
approx LB

Part B

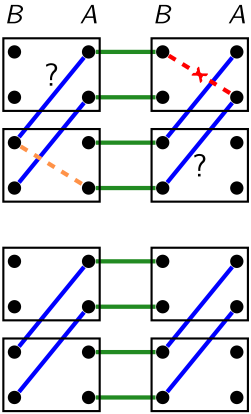


Three Round Proof Sketch II

Part A



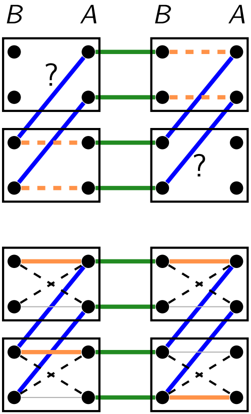
Part B



$0.625 + o(1)$
approx LB

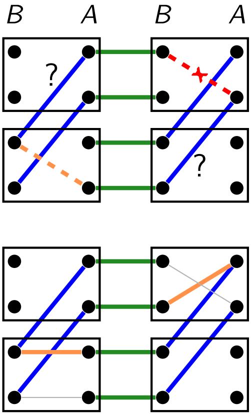
Three Round Proof Sketch II

Part A



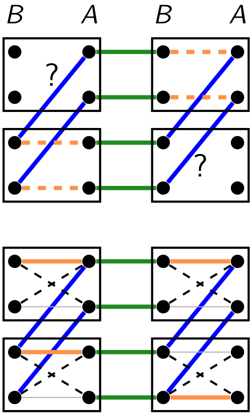
$0.625 + o(1)$
approx LB

Part B



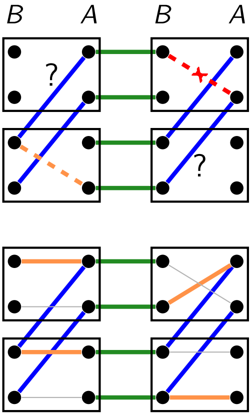
Three Round Proof Sketch II

Part A



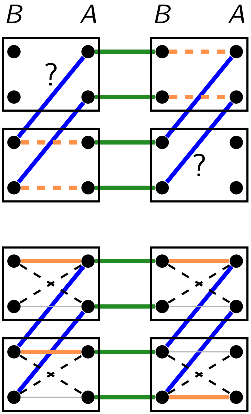
$0.625 + o(1)$
approx LB

Part B



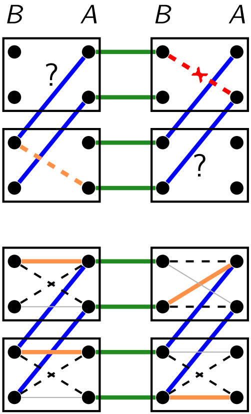
Three Round Proof Sketch II

Part A



$0.625 + o(1)$
approx LB

Part B



$0.625 + o(1)$
approx LB

Summary

Lower Bounds

There **does not exist** a deterministic algorithm for **MM** (even for **MBM**) in the edge query model that achieves a better than

1. 0.5-approximation in **1 round**,
2. $(0.5 + o(1))$-approximation in **2 rounds**, and
3. $(0.625 + o(1))$-approximation in **3 rounds**.

Summary

Lower Bounds

There **does not exist** a deterministic algorithm for **MM** (even for **MBM**) in the edge query model that achieves a better than

1. 0.5-approximation in **1 round**,
2. $(0.5 + o(1))$-approximation in **2 rounds**, and
3. $(0.625 + o(1))$-approximation in **3 rounds**.

Algorithm

0.625-approximation algorithm in **3-rounds** of the deterministic edge-query model.

Open Questions

Open Questions

Randomisation

Do randomised query algorithms allow us to improve on our results?

Open Questions

Randomisation

Do randomised query algorithms allow us to improve on our results?

Adaptivity

Can we obtain better query algorithms if we allow multiple non-adaptive queries per round?

Open Questions

Randomisation

Do randomised query algorithms allow us to improve on our results?

Adaptivity

Can we obtain better query algorithms if we allow multiple non-adaptive queries per round?

Semi-Streaming

Is there a 3-pass semi-streaming algorithms for **MBM** that improves on our 0.625-approximation algorithm?

Open Questions

Randomisation

Do randomised query algorithms allow us to improve on our results?

Adaptivity

Can we obtain better query algorithms if we allow multiple non-adaptive queries per round?

Semi-Streaming

Is there a 3-pass semi-streaming algorithms for **MBM** that improves on our 0.625-approximation algorithm?

Thank You!

References I

Sepehr Assadi, S. Cliff Liu, and Robert E. Tarjan, *An auction algorithm for bipartite matching in streaming and massively parallel computation models*, 4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021 (Hung Viet Le and Valerie King, eds.), SIAM, 2021, pp. 165–171.

Lidiya Khalidah binti Khalil and Christian Konrad, *Constructing large matchings via query access to a maximal matching oracle*, 40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2020, December 14-18, 2020, BITS Pilani, K K Birla Goa Campus, Goa, India (Virtual Conference) (Nitin Saxena and Sunil Simon, eds.), LIPIcs, vol. 182, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, pp. 26:1–26:15.

References II

- Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang, *On graph problems in a semi-streaming model*, Theor. Comput. Sci. **348** (2005), no. 2-3, 207–216.

- 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

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.

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.