On Two-Pass Streaming Algorithms for Maximum Bipartite Matching

Kheeran K. Naidu

University of Bristol

kn16063@bristol.ac.uk

Joint work with Dr Christian Konrad

Overview

1. Background
2. Our Work
   - Lower Bound
   - Algorithmic
3. Discussion

Matchings

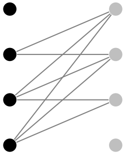
Definition

A **(bipartite) matching** is a subset of edges of a graph where every vertex has degree at most 1.

Matchings

Definition

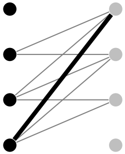
A **(bipartite) matching** is a subset of edges of a graph where every vertex has degree at most 1.



Matchings

Definition

A **(bipartite) matching** is a subset of edges of a graph where every vertex has degree at most 1.

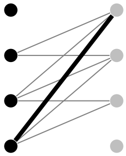


matching

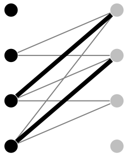
Matchings

Definition

A **(bipartite) matching** is a subset of edges of a graph where every vertex has degree at most 1.



matching

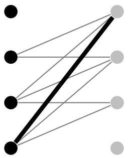


maximal matching

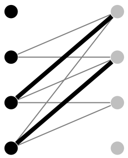
Matchings

Definition

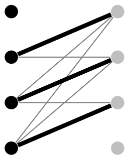
A **(bipartite) matching** is a subset of edges of a graph where every vertex has degree at most 1.



matching



maximal matching



maximum matching

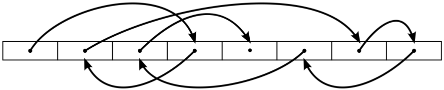
Matchings

Definition

A **(bipartite) matching** is a subset of edges of a graph where every vertex has degree at most 1.

Finding a maximum matching:

- exact algorithms exists, i.e. Hopcroft-Karp [SWAT71];
- require random access to the graph's edges (infeasible requirement for massive graphs).

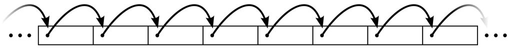


Semi-Streaming Model

Feigenbaum et al. [ICALP04]

A graph with n vertices is presented to an algorithm as a stream of edges where the storage space of the algorithm is bounded by $O(n \operatorname{polylog} n)$.

- Only allows **sequential access** to the graph.
- Algorithms with space $O(n \operatorname{polylog} n) = O(n\ (\log n)^{O(1)})$.
- Ideally with few passes of the stream.



Maximum Bipartite Matching Literature I

	Algorithmic	Lower Bound
one-pass	$\frac{1}{2}$ [folklore]	$\frac{1}{2} + 0.167$ [SODA12] $\frac{1}{2} + 0.132$ [SODA13] $\frac{1}{2} + 0.091$ [SODA21]
two-pass	$\frac{1}{2} + 0.019$ [APPROX12] $\frac{1}{2} + 0.083$ [ICDMW16] $\frac{1}{2} + 0.063$ [APPROX17] $\frac{1}{2} + 0.085$ [MFCS18]	

Main Results

Two-Pass Maximum Bipartite Matching

Class of algorithms:

1. finds a maximal matching M (first-pass);
2. increases the size of the matching M (second-pass).

Main Results

Two-Pass Maximum Bipartite Matching

Class of algorithms:

1. finds a maximal matching M (first-pass);
2. increases the size of the matching M (second-pass).

Note: Only strategy proven to work.

Two-Pass Maximum Bipartite Matching

Class of algorithms:

1. finds a maximal matching M (first-pass);
2. increases the size of the matching M (second-pass).

Note: Only strategy proven to work.

Lower Bound:

Algorithmic:

Main Results

Two-Pass Maximum Bipartite Matching

Class of algorithms:

1. finds a maximal matching M (first-pass);
2. increases the size of the matching M (second-pass).

Note: Only strategy proven to work.

Lower Bound:

- If the first pass finds only a maximal matching, no better than a $\frac{2}{3} \approx \frac{1}{2} + 0.167$-approximation is possible in two passes.

Algorithmic:

Main Results

Two-Pass Maximum Bipartite Matching

Class of algorithms:

1. finds a maximal matching M (first-pass);
2. increases the size of the matching M (second-pass).

Note: Only strategy proven to work.

Lower Bound:

- If the first pass finds only a maximal matching, no better than a $\frac{2}{3} \approx \frac{1}{2} + 0.167$-approximation is possible in two passes.

Algorithmic:

- Even a combination of the two dominant techniques in the area cannot beat the current state-of-the-art $2 - \sqrt{2} \approx \frac{1}{2} + 0.085$-approximation.

Main Results

Two-Pass Maximum Bipartite Matching

Class of algorithms:

1. finds a maximal matching M (first-pass);
2. increases the size of the matching M (second-pass).

Note: Only strategy proven to work.

Lower Bound:

- If the first pass finds only a maximal matching, no better than a $\frac{2}{3} \approx \frac{1}{2} + 0.167$-approximation is possible in two passes.

Algorithmic:

- Even a combination of the two dominant techniques in the area cannot beat the current state-of-the-art $2 - \sqrt{2} \approx \frac{1}{2} + 0.085$-approximation.

Other strategies and techniques are required!

Maximum Bipartite Matching Literature II

	Algorithmic	Lower Bound
one-pass	$\frac{1}{2}$ [folklore]	$\frac{1}{2} + 0.167$ [SODA12] $\frac{1}{2} + 0.132$ [SODA13] $\frac{1}{2} + 0.091$ [SODA21]
two-pass	$\frac{1}{2} + 0.019$ [APPROX12] $\frac{1}{2} + 0.083$ [ICDMW16] $\frac{1}{2} + 0.063$ [APPROX17] $\frac{1}{2} + 0.085$ [MFCS18] $\frac{1}{2} + 0.085$	$\frac{1}{2} + 0.167$[1]

[1] where the first pass finds a maximal matching, i.e., at least a $\frac{1}{2}$-approximation.

Overview

Overview

Lower Bound Result

Two-Pass Maximum Bipartite Matching

Lower Bound:

- If the first pass finds only a maximal matching, no better than a $\frac{2}{3} \approx \frac{1}{2} + 0.167$-approximation is possible in two passes.

Lower Bound Result

Two-Pass Maximum Bipartite Matching

Lower Bound:

- If the first pass finds only a maximal matching, no better than a $\frac{2}{3} \approx \frac{1}{2} + 0.167$-approximation is possible in two passes.
- Extends Goel, Kapralov and Khanna's one-pass $\frac{2}{3}$-approximation lower bound [SODA12].

Lower Bound Result

Two-Pass Maximum Bipartite Matching

Lower Bound:

- If the first pass finds only a maximal matching, no better than a $\frac{2}{3} \approx \frac{1}{2} + 0.167$-approximation is possible in two passes.
- Extends Goel, Kapralov and Khanna's one-pass $\frac{2}{3}$-approximation lower bound [SODA12].
- Uses a dense family of Rusza Szemeredi (RS) graphs which contains (many) near-perfect matchings.

Lower Bound Result

Two-Pass Maximum Bipartite Matching

Lower Bound:

- If the first pass finds only a maximal matching, no better than a $\frac{2}{3} \approx \frac{1}{2} + 0.167$-approximation is possible in two passes.
- Extends Goel, Kapralov and Khanna's one-pass $\frac{2}{3}$-approximation lower bound [SODA12].
- Uses a dense family of Rusza Szemeredi (RS) graphs which contains (many) near-perfect matchings. *This result may be of independent interest.*

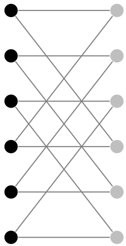
Rusza Szemeredi Graphs

Definition

Rusza Szemeredi (RS) graphs are a family of bipartite graphs whose edge set is made up of a union of induced matchings of the same size.

Definition

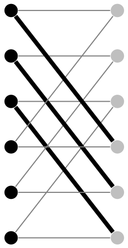
Rusza Szemeredi (RS) graphs are a family of bipartite graphs whose edge set is made up of a union of induced matchings of the same size.



Rusza Szemeredi Graphs

Definition

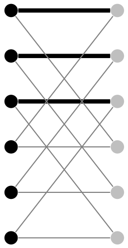
Rusza Szemeredi (RS) graphs are a family of bipartite graphs whose edge set is made up of a union of induced matchings of the same size.



Rusza Szemeredi Graphs

Definition

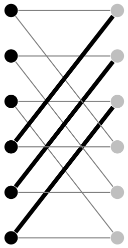
Rusza Szemeredi (RS) graphs are a family of bipartite graphs whose edge set is made up of a union of induced matchings of the same size.



Rusza Szemeredi Graphs

Definition

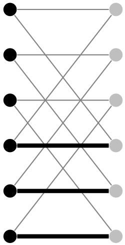
Rusza Szemeredi (RS) graphs are a family of bipartite graphs whose edge set is made up of a union of induced matchings of the same size.



Rusza Szemeredi Graphs

Definition

Rusza Szemeredi (RS) graphs are a family of bipartite graphs whose edge set is made up of a union of induced matchings of the same size.



Rusza Szemeredi Graphs

Definition

Rusza Szemeredi (RS) graphs are a family of bipartite graphs whose edge set is made up of a union of induced matchings of the same size.

Proposition ([SODA12])

There exists a bipartite RS graph with $N^{\Omega(\frac{1}{\log\log N})}$ induced matchings of size $\frac{N}{2} - \epsilon N$ where $N = |A| = |B|$ and $\epsilon > 0$.

Rusza Szemeredi Graphs

Definition

Rusza Szemeredi (RS) graphs are a family of bipartite graphs whose edge set is made up of a union of induced matchings of the same size.

Proposition ([SODA12])

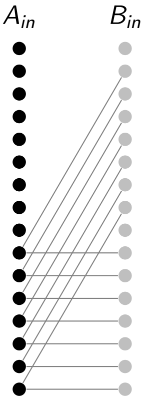
There exists a bipartite RS graph with $N^{\Omega(\frac{1}{\log\log N})}$ induced matchings of size $\frac{N}{2} - \epsilon N$ where $N = |A| = |B|$ and $\epsilon > 0$.

Note: This is a dense RS graph with $N^{1+\Omega(\frac{1}{\log\log N})} \supset O(N \operatorname{polylog} N)$ edges.

One-Pass Lower Bound Proof [SODA12]

Two-Party Communication Setup

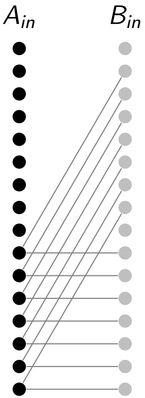
1. Start with the dense RS graph with $N^{\Omega(\frac{1}{\log \log N})}$ induced matchings of size $\frac{N}{2} - \epsilon N$.



One-Pass Lower Bound Proof [SODA12]

Two-Party Communication Setup

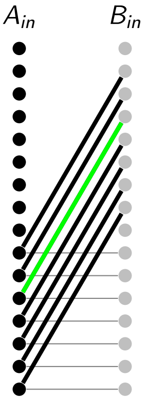
1. Start with the dense RS graph with $N^{\Omega(\frac{1}{\log \log N})}$ induced matchings of size $\frac{N}{2} - \epsilon N$.
2. Randomly remove ϵN edges from each induced matching.



One-Pass Lower Bound Proof [SODA12]

Two-Party Communication Setup

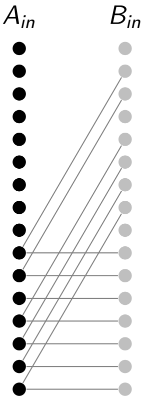
1. Start with the dense RS graph with $N^{\Omega(\frac{1}{\log \log N})}$ induced matchings of size $\frac{N}{2} - \epsilon N$.
2. Randomly remove ϵN edges from each induced matching.



One-Pass Lower Bound Proof [SODA12]

Two-Party Communication Setup

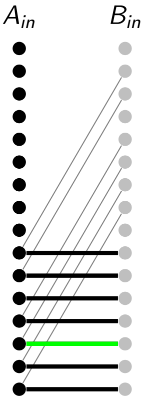
1. Start with the dense RS graph with $N^{\Omega(\frac{1}{\log\log N})}$ induced matchings of size $\frac{N}{2} - \epsilon N$.
2. Randomly remove ϵN edges from each induced matching.



One-Pass Lower Bound Proof [SODA12]

Two-Party Communication Setup

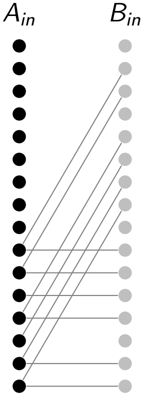
1. Start with the dense RS graph with $N^{\Omega(\frac{1}{\log\log N})}$ induced matchings of size $\frac{N}{2} - \epsilon N$.
2. Randomly remove ϵN edges from each induced matching.



One-Pass Lower Bound Proof [SODA12]

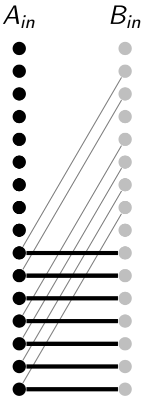
Two-Party Communication Setup

1. Start with the dense RS graph with $N^{\Omega(\frac{1}{\log\log N})}$ induced matchings of size $\frac{N}{2} - \epsilon N$.
2. Randomly remove ϵN edges from each induced matching.



Two-Party Communication Setup

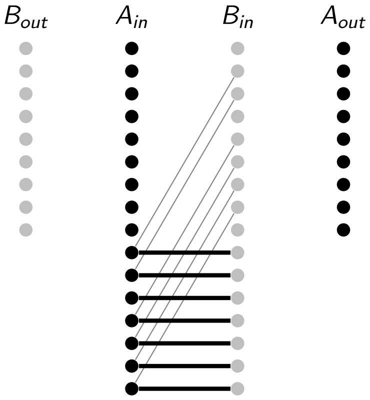
1. Start with the dense RS graph with $N^{\Omega(\frac{1}{\log\log N})}$ induced matchings of size $\frac{N}{2} - \epsilon N$.
2. Randomly remove ϵN edges from each induced matching.
3. Select a single special matching M_j at random.



One-Pass Lower Bound Proof [SODA12]

Two-Party Communication Setup

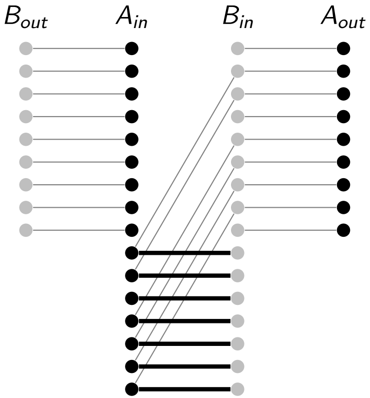
1. Start with the dense RS graph with $N^{\Omega(\frac{1}{\log \log N})}$ induced matchings of size $\frac{N}{2} - \epsilon N$.
2. Randomly remove ϵN edges from each induced matching.
3. Select a single special matching M_j at random.
4. Introduce two new sets of outer vertices each of size $\frac{N}{2} + \epsilon N$ and connect to them to the non-M_j inner vertices by a perfect matching.



One-Pass Lower Bound Proof [SODA12]

Two-Party Communication Setup

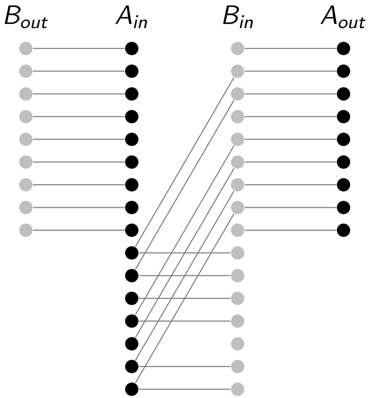
1. Start with the dense RS graph with $N^{\Omega(\frac{1}{\log\log N})}$ induced matchings of size $\frac{N}{2} - \epsilon N$.
2. Randomly remove ϵN edges from each induced matching.
3. Select a single special matching M_j at random.
4. Introduce two new sets of outer vertices each of size $\frac{N}{2} + \epsilon N$ and connect to them to the non-M_j inner vertices by a perfect matching.



One-Pass Lower Bound Proof [SODA12]

Two-Party Communication Setup

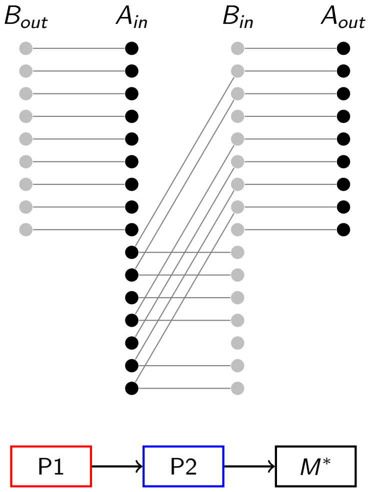
1. Start with the dense RS graph with $N^{\Omega(\frac{1}{\log\log N})}$ induced matchings of size $\frac{N}{2} - \epsilon N$.
2. Randomly remove ϵN edges from each induced matching.
3. Select a single special matching M_j at random.
4. Introduce two new sets of outer vertices each of size $\frac{N}{2} + \epsilon N$ and connect to them to the non-M_j inner vertices by a perfect matching.



One-Pass Lower Bound Proof [SODA12]

Two-Party Communication Setup

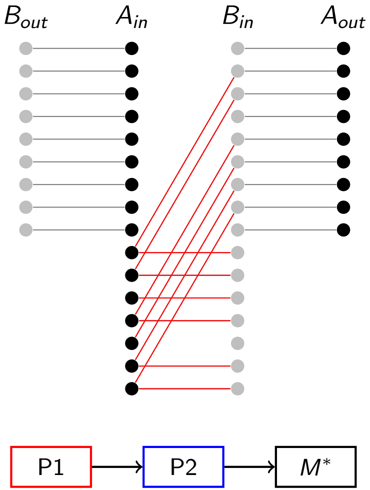
1. Start with the dense RS graph with $N^{\Omega(\frac{1}{\log \log N})}$ induced matchings of size $\frac{N}{2} - \epsilon N$.
2. Randomly remove ϵN edges from each induced matching.
3. Select a single special matching M_j at random.
4. Introduce two new sets of outer vertices each of size $\frac{N}{2} + \epsilon N$ and connect to them to the non-M_j inner vertices by a perfect matching.



One-Pass Lower Bound Proof [SODA12]

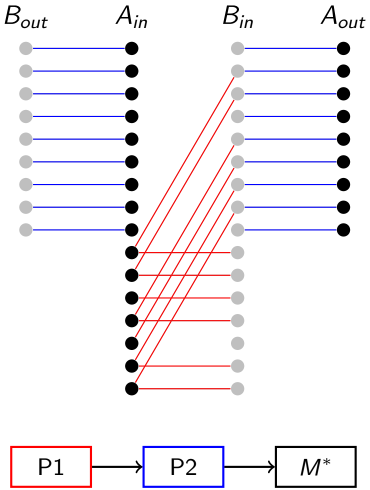
Two-Party Communication Setup

1. Start with the dense RS graph with $N^{\Omega(\frac{1}{\log\log N})}$ induced matchings of size $\frac{N}{2} - \epsilon N$.
2. Randomly remove ϵN edges from each induced matching.
3. Select a single special matching M_j at random.
4. Introduce two new sets of outer vertices each of size $\frac{N}{2} + \epsilon N$ and connect to them to the non-M_j inner vertices by a perfect matching.



Two-Party Communication Setup

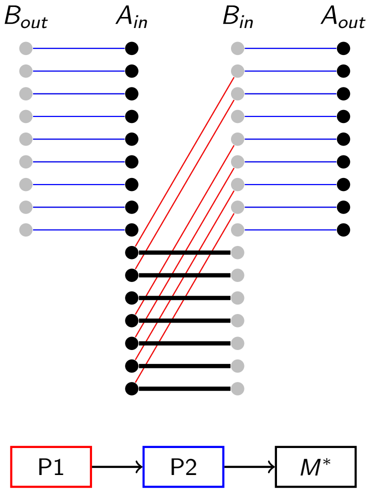
1. Start with the dense RS graph with $N^{\Omega(\frac{1}{\log\log N})}$ induced matchings of size $\frac{N}{2} - \epsilon N$.
2. Randomly remove ϵN edges from each induced matching.
3. Select a single special matching M_j at random.
4. Introduce two new sets of outer vertices each of size $\frac{N}{2} + \epsilon N$ and connect to them to the non-M_j inner vertices by a perfect matching.



One-Pass Lower Bound Proof [SODA12]

Two-Party Communication Setup

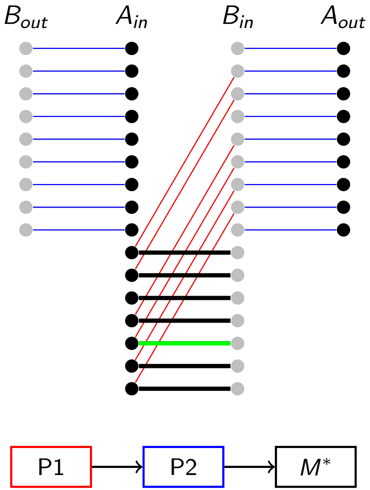
1. Start with the dense RS graph with $N^{\Omega(\frac{1}{\log\log N})}$ induced matchings of size $\frac{N}{2} - \epsilon N$.
2. Randomly remove ϵN edges from each induced matching.
3. Select a single special matching M_j at random.
4. Introduce two new sets of outer vertices each of size $\frac{N}{2} + \epsilon N$ and connect to them to the non-M_j inner vertices by a perfect matching.



One-Pass Lower Bound Proof [SODA12]

Two-Party Communication Setup

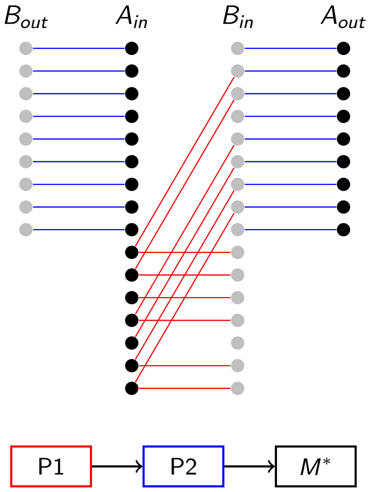
1. Start with the dense RS graph with $N^{\Omega(\frac{1}{\log\log N})}$ induced matchings of size $\frac{N}{2} - \epsilon N$.
2. Randomly remove ϵN edges from each induced matching.
3. Select a single special matching M_j at random.
4. Introduce two new sets of outer vertices each of size $\frac{N}{2} + \epsilon N$ and connect to them to the non-M_j inner vertices by a perfect matching.



One-Pass Lower Bound Proof [SODA12]

Two-Party Communication Setup

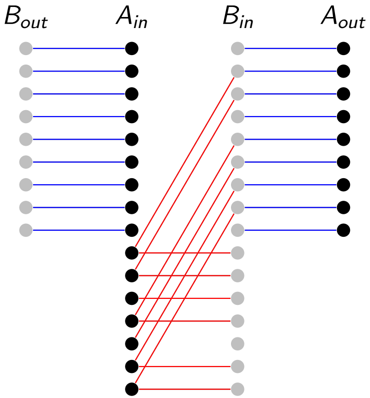
1. Start with the dense RS graph with $N^{\Omega(\frac{1}{\log \log N})}$ induced matchings of size $\frac{N}{2} - \epsilon N$.
2. Randomly remove ϵN edges from each induced matching.
3. Select a single special matching M_j at random.
4. Introduce two new sets of outer vertices each of size $\frac{N}{2} + \epsilon N$ and connect to them to the non-M_j inner vertices by a perfect matching.
5. $(\frac{2}{3} + \epsilon)$-approx

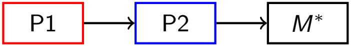


One-Pass Lower Bound Proof [SODA12]

Two-Party Communication Setup

1. Start with the dense RS graph with $N^{\Omega(\frac{1}{\log\log N})}$ induced matchings of size $\frac{N}{2} - \epsilon N$.
2. Randomly remove ϵN edges from each induced matching.
3. Select a single special matching M_j at random.
4. Introduce two new sets of outer vertices each of size $\frac{N}{2} + \epsilon N$ and connect to them to the non-M_j inner vertices by a perfect matching.
5. $(\frac{2}{3} + \epsilon)$-approx

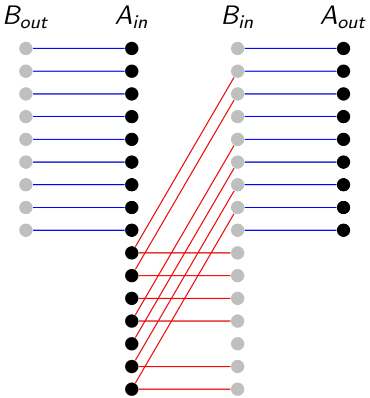




One-Pass Lower Bound Proof [SODA12]

Two-Party Communication Setup

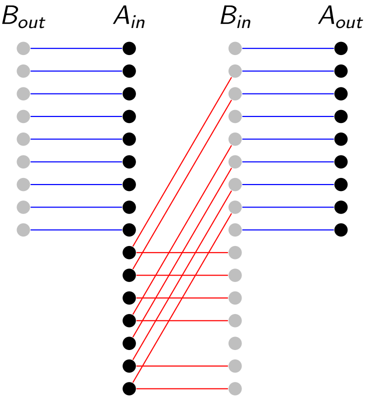
1. Start with the dense RS graph with $N^{\Omega(\frac{1}{\log\log N})}$ induced matchings of size $\frac{N}{2} - \epsilon N$.
2. Randomly remove ϵN edges from each induced matching.
3. Select a single special matching M_j at random.
4. Introduce two new sets of outer vertices each of size $\frac{N}{2} + \epsilon N$ and connect to them to the non-M_j inner vertices by a perfect matching.
5. $(\frac{2}{3} + \epsilon)$-approx requires space $N^{1+\Omega(\frac{1}{\log\log N})} \supset O(N \operatorname{polylog} N)$.

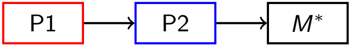


$$\boxed{P1} \longrightarrow \boxed{P2} \longrightarrow \boxed{M^*}$$

Two-Pass Lower Bound Proof

Goal: Give both players knowledge of a maximal matching without affecting the difficulty of the problem.



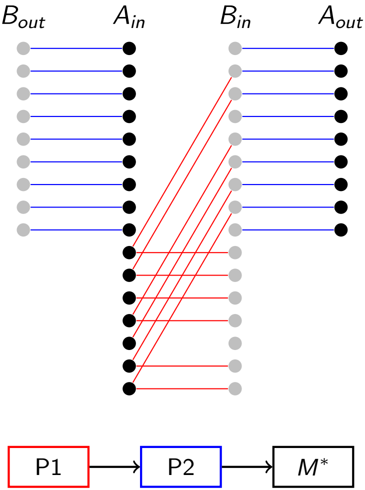


Two-Pass Lower Bound Proof

Goal: Give both players knowledge of a maximal matching without affecting the difficulty of the problem.

Outline:

- Do dense RS graphs contain perfect matchings?

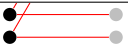


Two-Pass Lower Bound Proof

Goal: Give both players knowledge of a maximal matching without affecting the dif

Ou

B_{out} A_{in} B_{in} A_{out}



Proposition

There exists a bipartite RS graph with $N^{\Omega(\frac{1}{\log\log N})}$ induced matchings of size $\frac{N}{2} - \epsilon N$ where $N = |A| = |B|$ and $\epsilon > 0$ such that there are $N^{\Omega(\frac{1}{\log\log N})}$ disjoint near-perfect matchings, each of size $N - 2\epsilon N$.

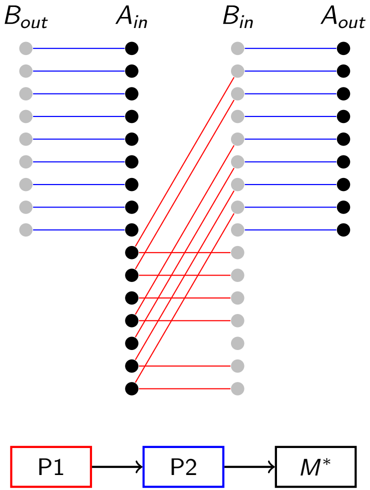
P1 $\longrightarrow$ P2 $\longrightarrow$ M^*

Two-Pass Lower Bound Proof

Goal: Give both players knowledge of a maximal matching without affecting the difficulty of the problem.

Outline:

- Do dense RS graphs contain perfect matchings?

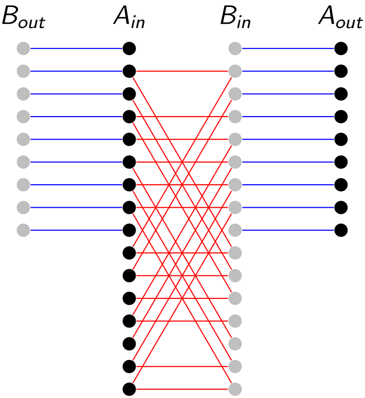


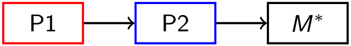
Two-Pass Lower Bound Proof

Goal: Give both players knowledge of a maximal matching without affecting the difficulty of the problem.

Outline:

- Do dense RS graphs contain perfect matchings?
- We use this RS graphs which has a near-perfect matching of size $N - \epsilon' N$ in the construction.



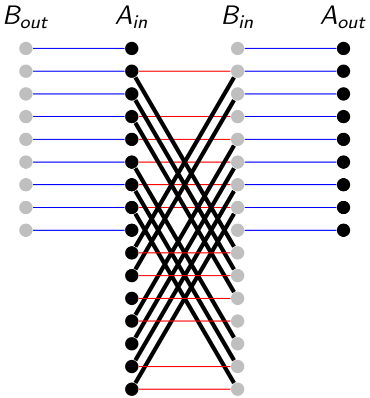


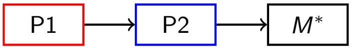
Two-Pass Lower Bound Proof

Goal: Give both players knowledge of a maximal matching without affecting the difficulty of the problem.

Outline:

- Do dense RS graphs contain perfect matchings?
- We use this RS graphs which has a near-perfect matching of size $N - \epsilon' N$ in the construction.

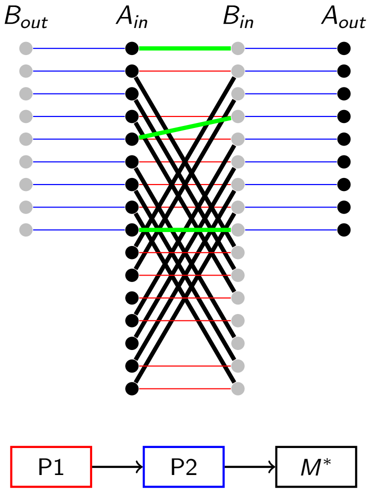




Goal: Give both players knowledge of a maximal matching without affecting the difficulty of the problem.

Outline:

- Do dense RS graphs contain perfect matchings?
- We use this RS graphs which has a near-perfect matching of size $N - \epsilon' N$ in the construction.

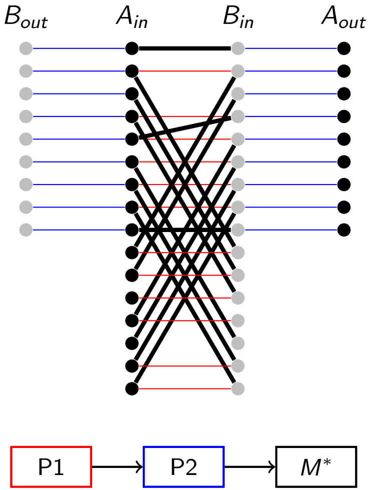


Two-Pass Lower Bound Proof

Goal: Give both players knowledge of a maximal matching without affecting the difficulty of the problem.

Outline:

- Do dense RS graphs contain perfect matchings?
- We use this RS graphs which has a near-perfect matching of size $N - \epsilon' N$ in the construction.
- $(\frac{2}{3} + \epsilon)$-approx requires space $N^{1+\Omega(\frac{1}{\log \log N})} \supset O(N \operatorname{polylog} N)$.



Overview

1. Background
2. Our Work
   - Lower Bound
   - Algorithmic
3. Discussion

Algorithmic Result

Two-Pass Maximum Bipartite Matching

Algorithmic:

- Even a combination of the two dominant techniques in the area cannot beat the current state-of-the-art $2 - \sqrt{2} \approx \frac{1}{2} + 0.085$-approximation.

Algorithmic Result

Two-Pass Maximum Bipartite Matching

Algorithmic:

- Even a combination of the two dominant techniques in the area cannot beat the current state-of-the-art $2 - \sqrt{2} \approx \frac{1}{2} + 0.085$-approximation.
- A novel meta algorithm that exactly achieves the current state-of-the-art.

Algorithmic Result

Two-Pass Maximum Bipartite Matching

Algorithmic:

- Even a combination of the two dominant techniques in the area cannot beat the current state-of-the-art $2 - \sqrt{2} \approx \frac{1}{2} + 0.085$-approximation.
- A novel meta algorithm that exactly achieves the current state-of-the-art.
- A family of hard-instance graphs shows the analysis is tight.

Algorithmic Approach

Class of algorithms:

1. finds a maximal matching M (first-pass);
2. increases the size of the matching M (second-pass).

Algorithmic Approach

Class of algorithms:

1. finds a maximal matching M (first-pass);
2. increases the size of the matching M (second-pass).

Definition

An **augmenting path** begins and ends with a vertex unmatched by the matching, and alternates along the path between edges in and out of the matching.

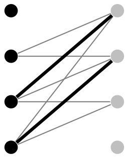
Algorithmic Approach

Class of algorithms:

1. finds a maximal matching M (first-pass);
2. increases the size of the matching M (second-pass).

Definition

An **augmenting path** begins and ends with a vertex unmatched by the matching, and alternates along the path between edges in and out of the matching.



$|M_0| = 2$

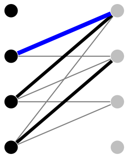
Algorithmic Approach

Class of algorithms:

1. finds a maximal matching M (first-pass);
2. increases the size of the matching M (second-pass).

Definition

An **augmenting path** begins and ends with a vertex unmatched by the matching, and alternates along the path between edges in and out of the matching.



$|M_0| = 2$

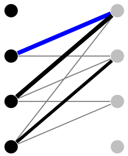
Algorithmic Approach

Class of algorithms:

1. finds a maximal matching M (first-pass);
2. increases the size of the matching M (second-pass).

Definition

An **augmenting path** begins and ends with a vertex unmatched by the matching, and alternates along the path between edges in and out of the matching.



$|M_0| = 2$

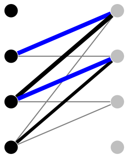
Algorithmic Approach

Class of algorithms:

1. finds a maximal matching M (first-pass);
2. increases the size of the matching M (second-pass).

Definition

An **augmenting path** begins and ends with a vertex unmatched by the matching, and alternates along the path between edges in and out of the matching.



$|M_0| = 2$

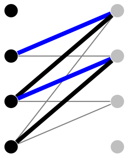
Algorithmic Approach

Class of algorithms:

1. finds a maximal matching M (first-pass);
2. increases the size of the matching M (second-pass).

Definition

An **augmenting path** begins and ends with a vertex unmatched by the matching, and alternates along the path between edges in and out of the matching.



$|M_0| = 2$

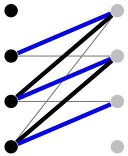
Algorithmic Approach

Class of algorithms:

1. finds a maximal matching M (first-pass);
2. increases the size of the matching M (second-pass).

Definition

An **augmenting path** begins and ends with a vertex unmatched by the matching, and alternates along the path between edges in and out of the matching.



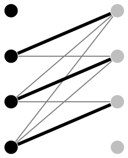
$|M_0| = 2$

Class of algorithms:

1. finds a maximal matching M (first-pass);
2. increases the size of the matching M (second-pass).

Definition

An **augmenting path** begins and ends with a vertex unmatched by the matching, and alternates along the path between edges in and out of the matching.

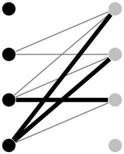


$|M_1| = 3$

Two-Pass Algorithm Summary

Using the two dominant techniques, parameterised by p and d:

1. subsampling with probability p [APPROX12] [MFCS18]
2. run $\textsc{Greedy}_d$ [ICDMW16] [APPROX17]

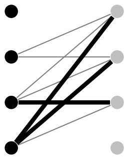


semi-incomplete matching

Two-Pass Algorithm Summary

Using the two dominant techniques, parameterised by p and d:

1. subsampling with probability p [APPROX12] [MFCS18]
2. run $\textsc{Greedy}_d$ [ICDMW16] [APPROX17]



semi-incomplete matching

Let $G = (A, B, E)$ be any bipartite graph.

Let $\pi = e_1, e_2, \ldots$ be any stream of its edges.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$

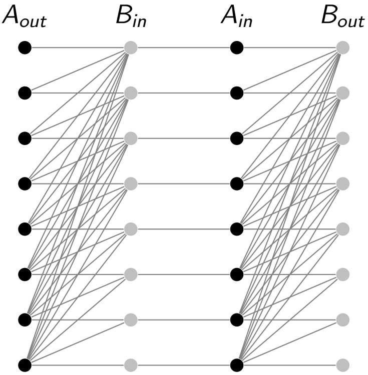
Second pass:

- subsample $M' \subseteq M$ with prob. p
- $H_L \leftarrow E \cap A_{out} \times B(M')$
- $H_R \leftarrow E \cap A(M') \times B_{out}$
- $M'_L \leftarrow \textsc{Greedy}_d(\pi_{H_L})$
- $M'_R \leftarrow \textsc{Greedy}_d(\pi_{H_R})$

Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

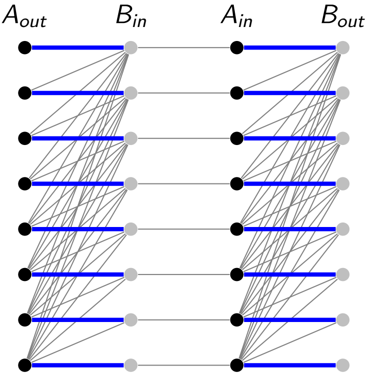


Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

The goal of the algorithm is to find the maximum matching M^*.



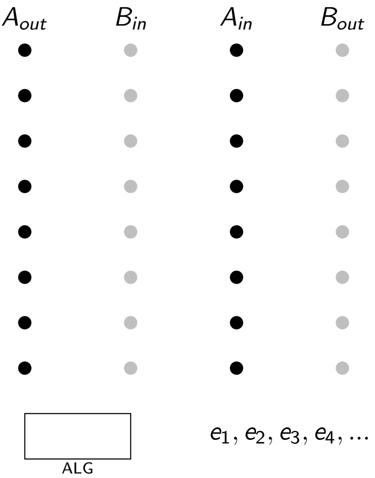
Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$



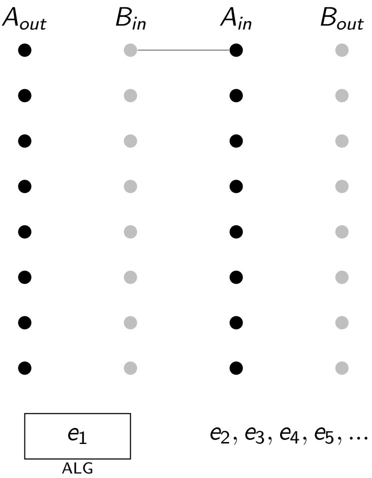
Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$



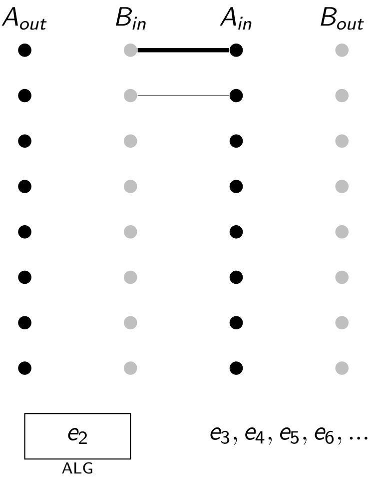
Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$

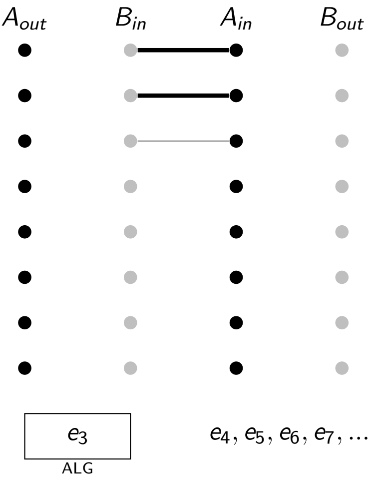


Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$



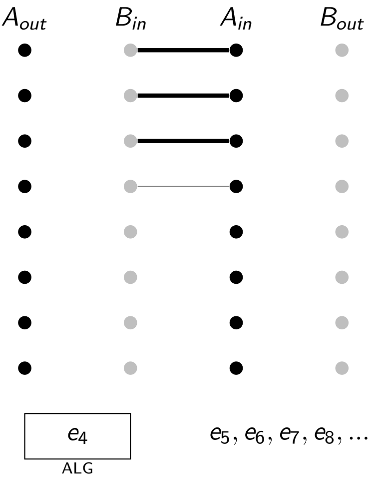
Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$



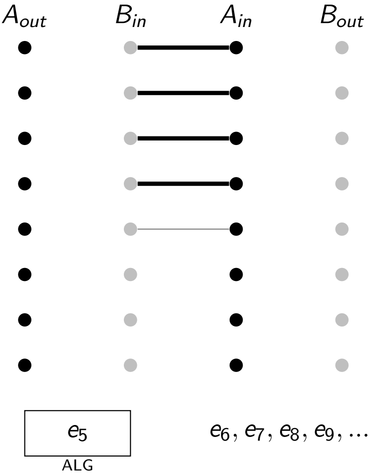
Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$



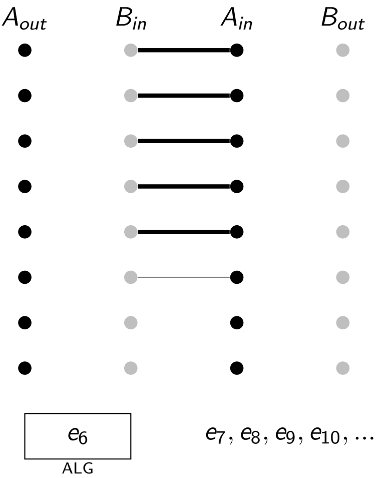
Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$



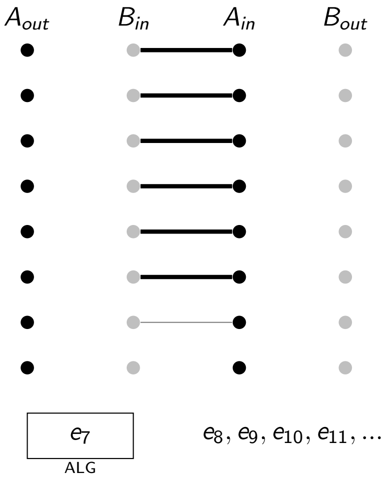
Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$



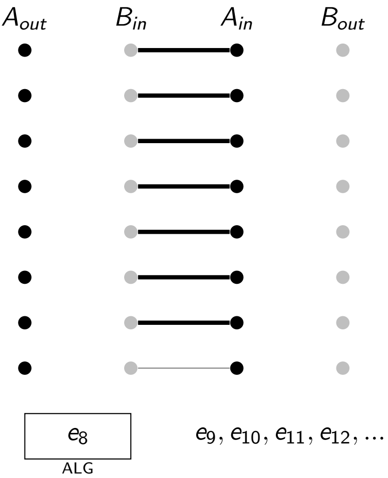
Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$



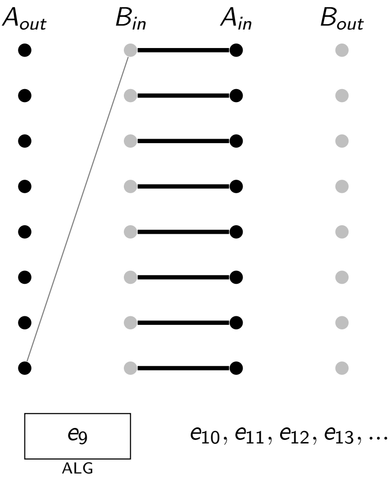
Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$



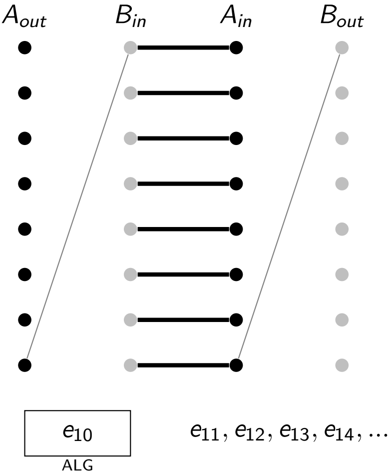
Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$



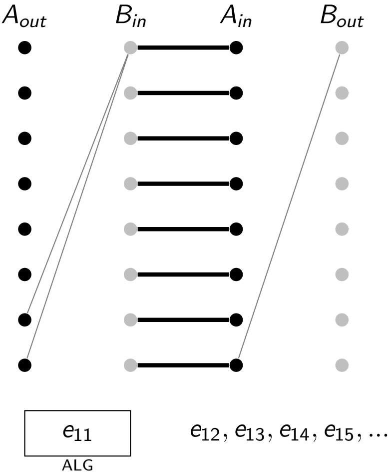
Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$



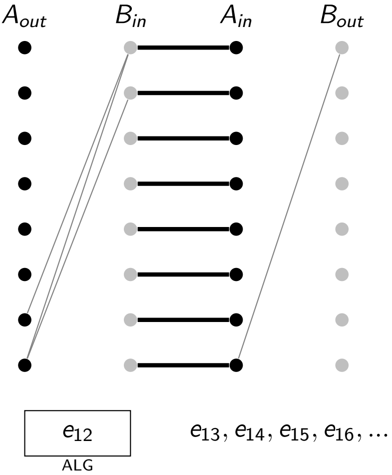
Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$



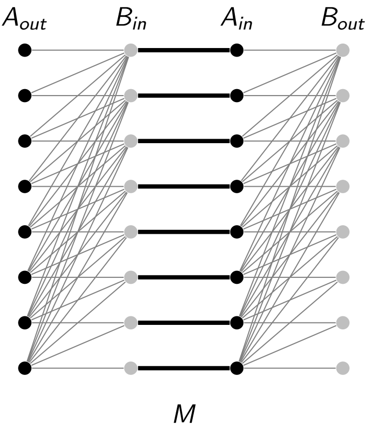
Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$



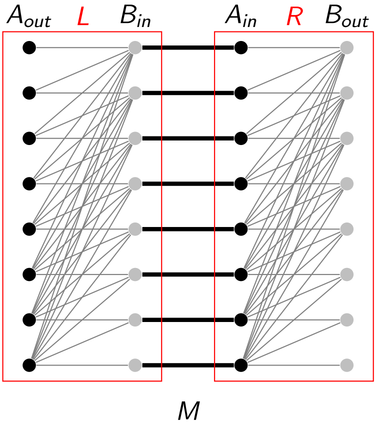
Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$



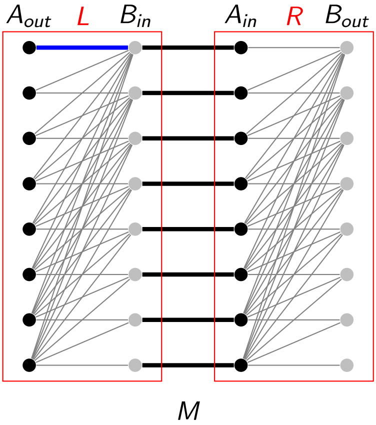
Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$



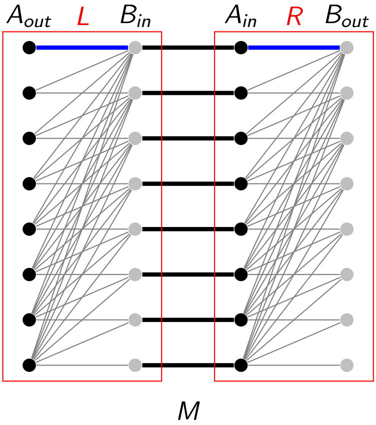
Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$



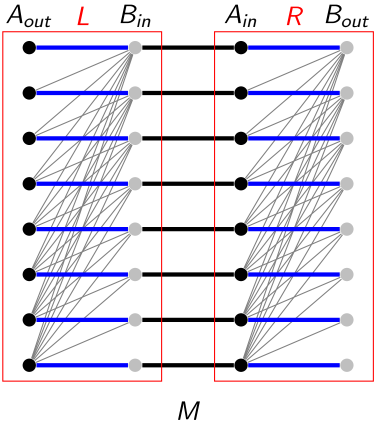
Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$



Two-Pass Algorithm

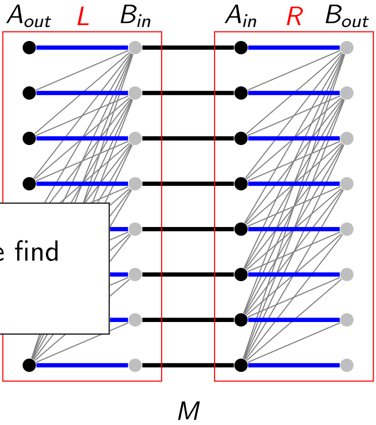
Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$

How can we find these?



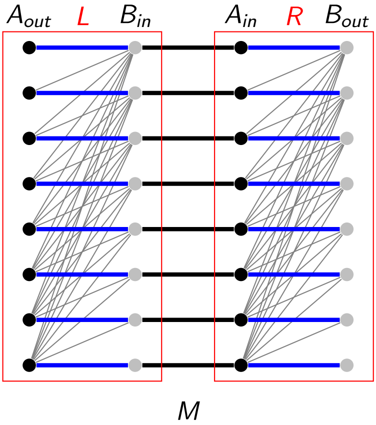
Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$



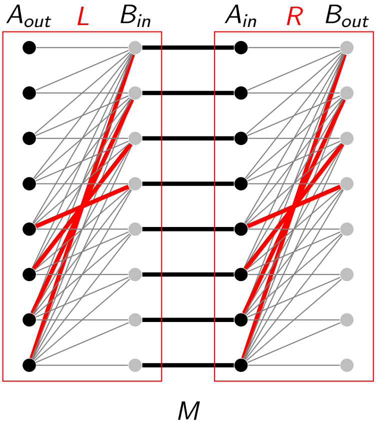
Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$



Two-Pass Algorithm

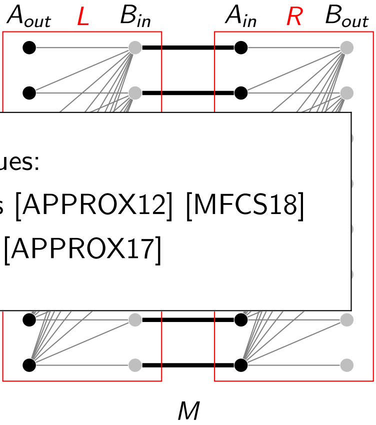
Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edge

order

First

-



Using the two dominant techniques:

1. subsample the inner vertices [APPROX12] [MFCS18]
2. run $\textsc{Greedy}_d$ [ICDMW16] [APPROX17]

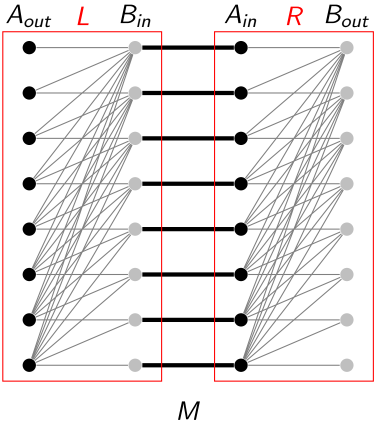
Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$



Two-Pass Algorithm

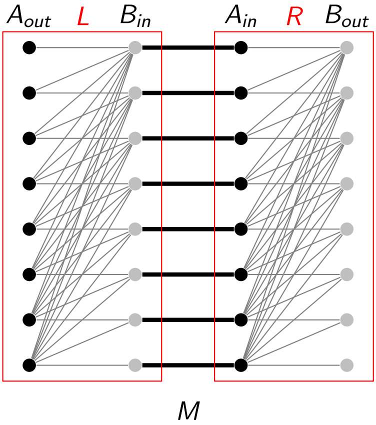
Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$

Second pass:



Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

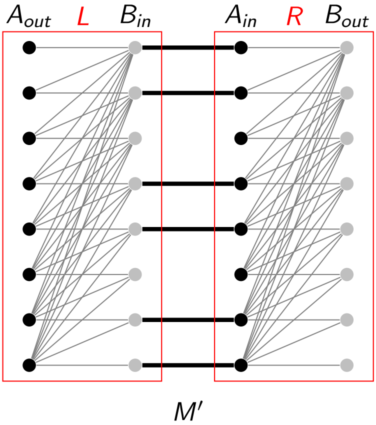
Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$

Second pass:

- subsample $M' \subseteq M$ with prob. p



$d = 3, \ p = 0.67$

Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

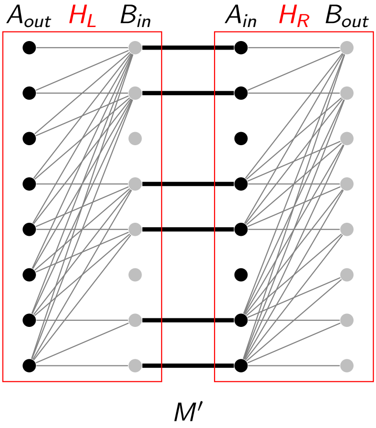
Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$

Second pass:

- subsample $M' \subseteq M$ with prob. p
- $H_L \leftarrow E \cap A_{out} \times B(M')$
- $H_R \leftarrow E \cap A(M') \times B_{out}$



$d = 3,\ p = 0.67$

Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

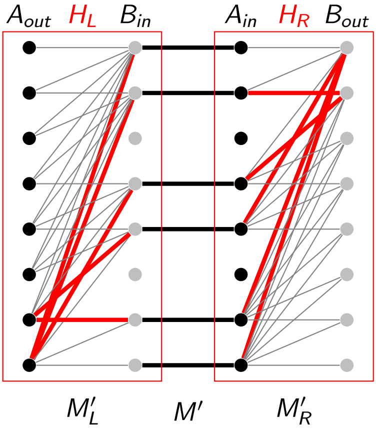
Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$

Second pass:

- subsample $M' \subseteq M$ with prob. p
- $H_L \leftarrow E \cap A_{out} \times B(M')$
- $H_R \leftarrow E \cap A(M') \times B_{out}$
- $M'_L \leftarrow \textsc{Greedy}_d(\pi_{H_L})$
- $M'_R \leftarrow \textsc{Greedy}_d(\pi_{H_R})$



$d = 3,\ p = 0.67$

Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

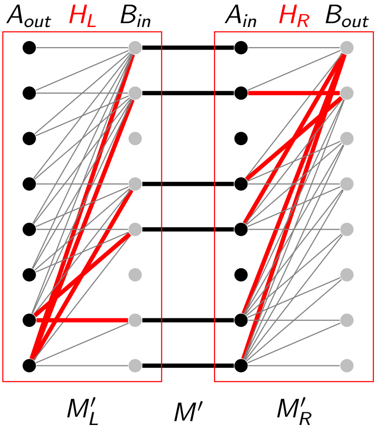
Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$

Second pass:

- subsample $M' \subseteq M$ with prob. p
- $H_L \leftarrow E \cap A_{out} \times B(M')$
- $H_R \leftarrow E \cap A(M') \times B_{out}$
- $M'_L \leftarrow \textsc{Greedy}_d(\pi_{H_L})$
- $M'_R \leftarrow \textsc{Greedy}_d(\pi_{H_R})$



$d = 3,\ p = 0.67$

$\mathbb{E}_{M'}[|M'_L|] = \frac{dp}{d+p} \cdot |M^*_L|$

Two-Pass Algorithm

Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

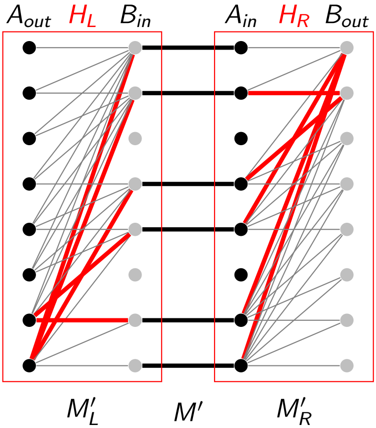
Let $\pi = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \textsc{Greedy}(\pi)$

Second pass:

- subsample $M' \subseteq M$ with prob. p
- $H_L \leftarrow E \cap A_{out} \times B(M')$
- $H_R \leftarrow E \cap A(M') \times B_{out}$
- $M'_L \leftarrow \textsc{Greedy}_d(\pi_{H_L})$
- $M'_R \leftarrow \textsc{Greedy}_d(\pi_{H_R})$



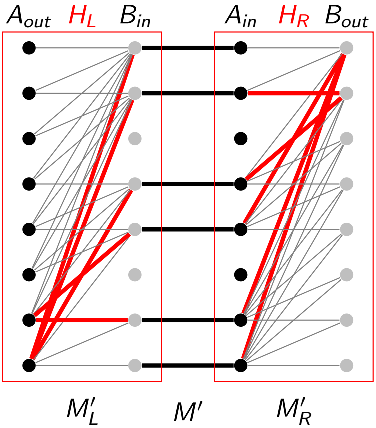
$d = 3,\ p = 0.67$

$$\mathbb{E}_{M'}[|M'_L|] = \frac{dp}{d+p} \cdot |M^*_L|$$

$$\mathbb{E}_{M'}[|M'_R|] = \frac{dp}{d+p} \cdot |M^*_R|$$

Two-Pass Algorithm

Analysis



$d = 3,\ p = 0.67$

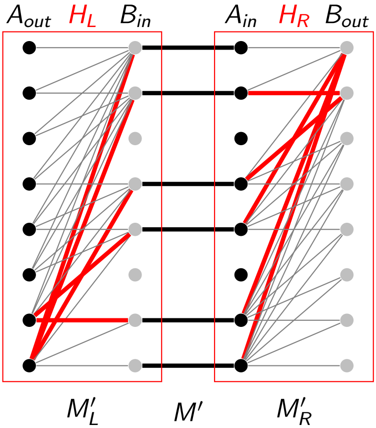
$$\mathbb{E}_{M'}[|M'_L|] = \frac{dp}{d+p} \cdot |M^*_L|$$

$$\mathbb{E}_{M'}[|M'_R|] = \frac{dp}{d+p} \cdot |M^*_R|$$

Two-Pass Algorithm

Analysis

$$|\mathcal{Q}| = (\frac{p}{d+p} - \frac{p}{2d}) \cdot |M^*|.$$



$d = 3,\ p = 0.67$

$$\mathbb{E}_{M'}[|M'_L|] = \frac{dp}{d+p} \cdot |M^*_L|$$

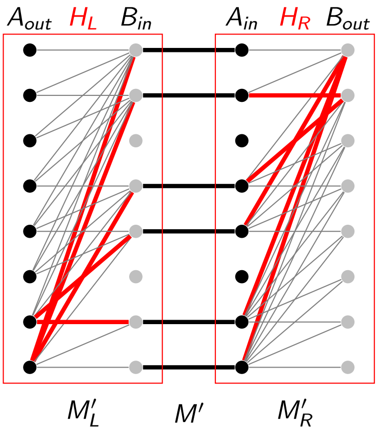
$$\mathbb{E}_{M'}[|M'_R|] = \frac{dp}{d+p} \cdot |M^*_R|$$

Analysis

$$|\mathcal{Q}| = \left(\frac{p}{d+p} - \frac{p}{2d}\right) \cdot |M^*|.$$

Therefore, the final matching is of size

$$\left(\frac{1}{2} + \frac{p}{d+p} - \frac{p}{2d}\right) \cdot |M^*|.$$



$d = 3,\ p = 0.67$

$$\mathbb{E}_{M'}[|M'_L|] = \tfrac{dp}{d+p} \cdot |M^*_L|$$

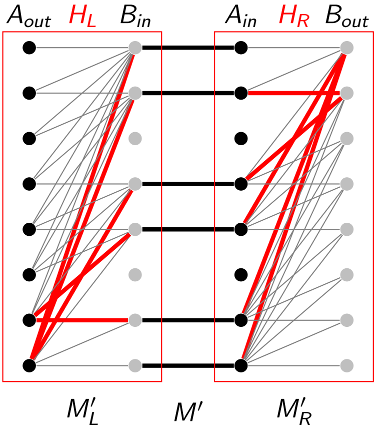
$$\mathbb{E}_{M'}[|M'_R|] = \tfrac{dp}{d+p} \cdot |M^*_R|$$

Analysis

$$|\mathcal{Q}| \geq (\frac{p}{d+p} - \frac{p}{2d}) \cdot |M^*|.$$

Therefore, the final matching is of size *at least*

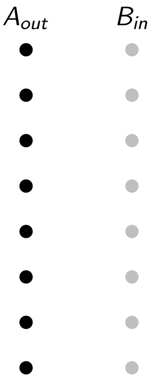
$$(\frac{1}{2} + \frac{p}{d+p} - \frac{p}{2d}) \cdot |M^*|.$$



$d = 3,\ p = 0.67$

$$\mathbb{E}_{M'}[|M'_L|] \geq \frac{dp}{d+p} \cdot |M^*_L|$$
$$\mathbb{E}_{M'}[|M'_R|] \geq \frac{dp}{d+p} \cdot |M^*_R|$$

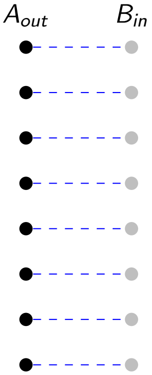
Main Proof Outline



Main Proof Outline

Setup:

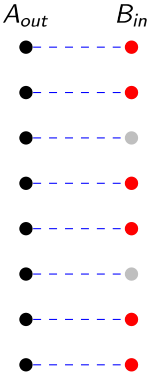
- any bipartite graph $G = (A, B, E)$ with a maximum matching M^* and any stream of edges π;



Main Proof Outline

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum matching M^* and any stream of edges π;
- subsample $B' \subseteq B$ with prob. p to get $H = G[A \cup B']$;

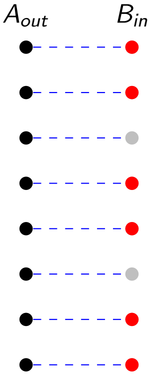


$d = 3,\ p = 0.67$

Main Proof Outline

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum matching M^* and any stream of edges π;
- subsample $B' \subseteq B$ with prob. p to get $H = G[A \cup B']$;
- $M \leftarrow \textsc{Greedy}_d(\pi_H)$.

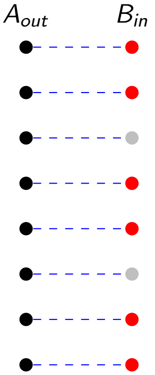


$d = 3,\ p = 0.67$

Main Proof Outline

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum matching M^* and any stream of edges π;
- subsample $B' \subseteq B$ with prob. p to get $H = G[A \cup B']$;
- $M \leftarrow \textsc{Greedy}_d(\pi_H)$.



$d = 3,\ p = 0.67$

Proof:

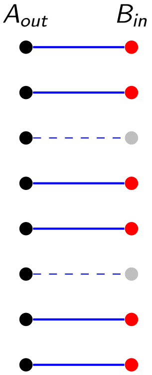
Main Proof Outline

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum matching M^* and any stream of edges π;
- subsample $B' \subseteq B$ with prob. p to get $H = G[A \cup B']$;
- $M \leftarrow \textsc{Greedy}_d(\pi_H)$.

Proof:

- $M^*_{B'} \leftarrow \{ab \in M^* : b \in B'\}$;



$d = 3,\ p = 0.67$

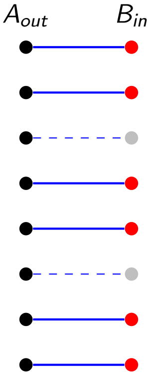
Main Proof Outline

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum matching M^* and any stream of edges π;
- subsample $B' \subseteq B$ with prob. p to get $H = G[A \cup B']$;
- $M \leftarrow \textsc{Greedy}_d(\pi_H)$.

Proof:

- $M^*_{B'} \leftarrow \{ab \in M^* : b \in B'\}$;
- For every edge that is added by $\textsc{Greedy}_d$, how many edges in $M^*_{B'}$ are blocked?



$d = 3,\ p = 0.67$

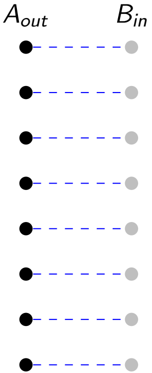
Main Proof Outline

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum matching M^* and any stream of edges π;
- subsample $B' \subseteq B$ with prob. p to get $H = G[A \cup B']$;
- $M \leftarrow \textsc{Greedy}_d(\pi_H)$.

Proof:

- $M^*_{B'} \leftarrow \{ab \in M^* : b \in B'\}$;
- For every edge that is added by $\textsc{Greedy}_d$, how many edges in $M^*_{B'}$ are blocked?



$d = 3$, $p = 0.67$

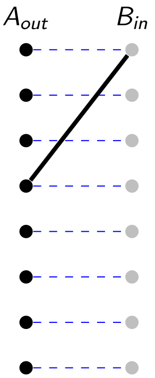
Main Proof Outline

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum matching M^* and any stream of edges π;
- subsample $B' \subseteq B$ with prob. p to get $H = G[A \cup B']$;
- $M \leftarrow \textsc{Greedy}_d(\pi_H)$.

Proof:

- $M^*_{B'} \leftarrow \{ab \in M^* : b \in B'\}$;
- For every edge that is added by $\textsc{Greedy}_d$, how many edges in $M^*_{B'}$ are blocked?



$d = 3,\ p = 0.67$

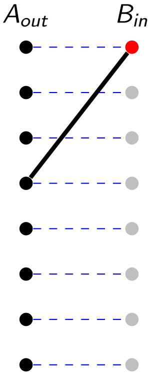
Main Proof Outline

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum matching M^* and any stream of edges π;
- subsample $B' \subseteq B$ with prob. p to get $H = G[A \cup B']$;
- $M \leftarrow \textsc{Greedy}_d(\pi_H)$.

Proof:

- $M^*_{B'} \leftarrow \{ab \in M^* : b \in B'\}$;
- For every edge that is added by $\textsc{Greedy}_d$, how many edges in $M^*_{B'}$ are blocked?



$d = 3$, $p = 0.67$

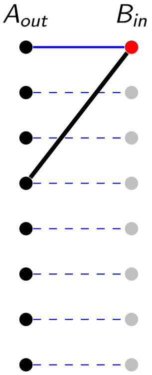
Main Proof Outline

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum matching M^* and any stream of edges π;
- subsample $B' \subseteq B$ with prob. p to get $H = G[A \cup B']$;
- $M \leftarrow \textsc{Greedy}_d(\pi_H)$.

Proof:

- $M^*_{B'} \leftarrow \{ab \in M^* : b \in B'\}$;
- For every edge that is added by $\textsc{Greedy}_d$, how many edges in $M^*_{B'}$ are blocked?



$d = 3,\ p = 0.67$

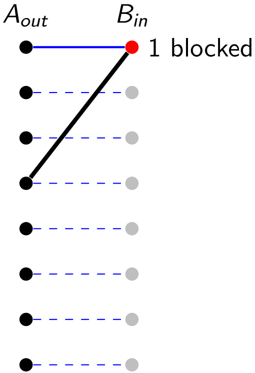
Main Proof Outline

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum matching M^* and any stream of edges π;
- subsample $B' \subseteq B$ with prob. p to get $H = G[A \cup B']$;
- $M \leftarrow \textsc{Greedy}_d(\pi_H)$.

Proof:

- $M^*_{B'} \leftarrow \{ab \in M^* : b \in B'\}$;
- For every edge that is added by $\textsc{Greedy}_d$, how many edges in $M^*_{B'}$ are blocked?



$d = 3,\ p = 0.67$

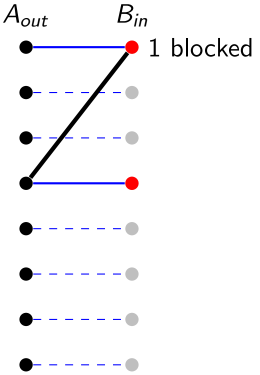
Main Proof Outline

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum matching M^* and any stream of edges π;
- subsample $B' \subseteq B$ with prob. p to get $H = G[A \cup B']$;
- $M \leftarrow \textsc{Greedy}_d(\pi_H)$.

Proof:

- $M^*_{B'} \leftarrow \{ab \in M^* : b \in B'\}$;
- For every edge that is added by $\textsc{Greedy}_d$, how many edges in $M^*_{B'}$ are blocked?



$d = 3$, $p = 0.67$

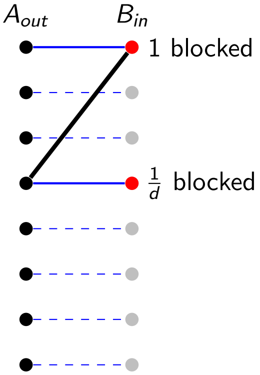
Main Proof Outline

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum matching M^* and any stream of edges π;
- subsample $B' \subseteq B$ with prob. p to get $H = G[A \cup B']$;
- $M \leftarrow \textsc{Greedy}_d(\pi_H)$.

Proof:

- $M^*_{B'} \leftarrow \{ab \in M^* : b \in B'\}$;
- For every edge that is added by $\textsc{Greedy}_d$, how many edges in $M^*_{B'}$ are blocked?



$d = 3$, $p = 0.67$

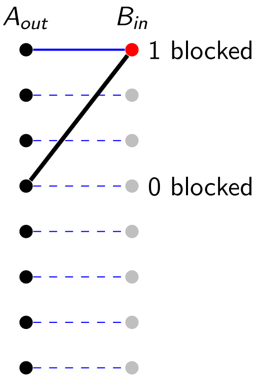
Main Proof Outline

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum matching M^* and any stream of edges π;
- subsample $B' \subseteq B$ with prob. p to get $H = G[A \cup B']$;
- $M \leftarrow \textsc{Greedy}_d(\pi_H)$.

Proof:

- $M^*_{B'} \leftarrow \{ab \in M^* : b \in B'\}$;
- For every edge that is added by $\textsc{Greedy}_d$, how many edges in $M^*_{B'}$ are blocked?



$d = 3$, $p = 0.67$

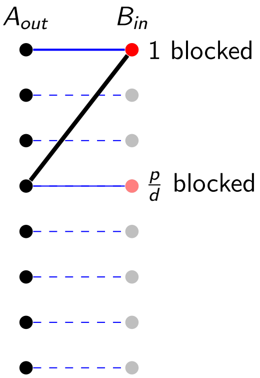
Main Proof Outline

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum matching M^* and any stream of edges π;
- subsample $B' \subseteq B$ with prob. p to get $H = G[A \cup B']$;
- $M \leftarrow \textsc{Greedy}_d(\pi_H)$.

Proof:

- $M^*_{B'} \leftarrow \{ab \in M^* : b \in B'\}$;
- For every edge that is added by $\textsc{Greedy}_d$, how many edges in $M^*_{B'}$ are blocked?



$d = 3$, $p = 0.67$

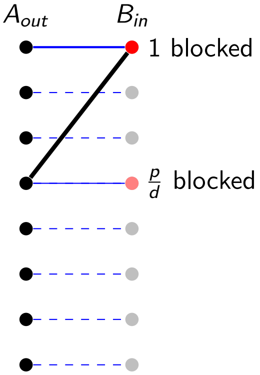
Main Proof Outline

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum matching M^* and any stream of edges π;
- subsample $B' \subseteq B$ with prob. p to get $H = G[A \cup B']$;
- $M \leftarrow \textsc{Greedy}_d(\pi_H)$.

Proof:

- $M^*_{B'} \leftarrow \{ab \in M^* : b \in B'\}$;
- For every edge that is added by $\textsc{Greedy}_d$, how many edges in $M^*_{B'}$ are blocked?



$d = 3,\ p = 0.67$

$$\mathbb{E}[|M^*_{B'}|] \leq (1 + \tfrac{p}{d})\mathbb{E}[|M|]$$

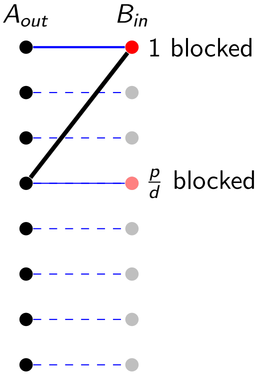
Main Proof Outline

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum matching M^* and any stream of edges π;
- subsample $B' \subseteq B$ with prob. p to get $H = G[A \cup B']$;
- $M \leftarrow \textsc{Greedy}_d(\pi_H)$.

Proof:

- $M^*_{B'} \leftarrow \{ab \in M^* : b \in B'\}$;
- For every edge that is added by $\textsc{Greedy}_d$, how many edges in $M^*_{B'}$ are blocked?



$d = 3,\ p = 0.67$

$p \cdot |M^*| \leq (1 + \frac{p}{d})\mathbb{E}[|M|]$

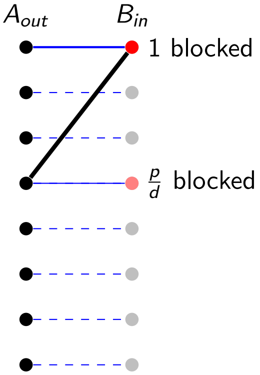
Main Proof Outline

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum matching M^* and any stream of edges π;
- subsample $B' \subseteq B$ with prob. p to get $H = G[A \cup B']$;
- $M \leftarrow \textsc{Greedy}_d(\pi_H)$.

Proof:

- $M^*_{B'} \leftarrow \{ab \in M^* : b \in B'\}$;
- For every edge that is added by $\textsc{Greedy}_d$, how many edges in $M^*_{B'}$ are blocked?



$d = 3,\ p = 0.67$

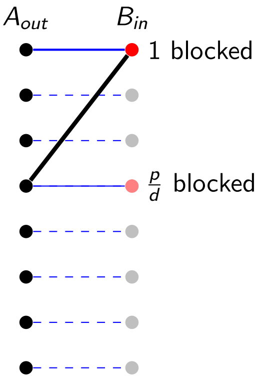
$\mathbb{E}[|M|] \geq \frac{dp}{d+p} \cdot |M^*|$

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum matching M^* and any stream of edges π;
- subsample $B' \subseteq B$ with prob. p to get $H = G[A \cup B']$;
- $M \leftarrow \textsc{Greedy}_d(\pi_H)$.

Proof:

- $M^*_{B'} \leftarrow \{ab \in M^* : b \in B'\}$;
- For every edge that is added by $\textsc{Greedy}_d$, how many edges in $M^*_{B'}$ are blocked?
- Formalised using Wald's Equation.



$d = 3,\ p = 0.67$

$$\mathbb{E}[|M|] \geq \frac{dp}{d+p} \cdot |M^*|$$

Optimal Algorithms

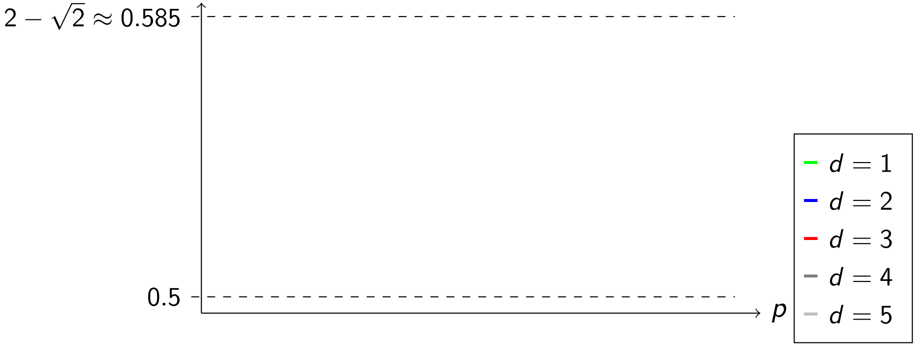
The final matching returned is always at least of size

$$(\frac{1}{2} + \frac{p}{d+p} - \frac{p}{2d}) \cdot |M^*|.$$

Optimal Algorithms

The final matching returned is always at least of size

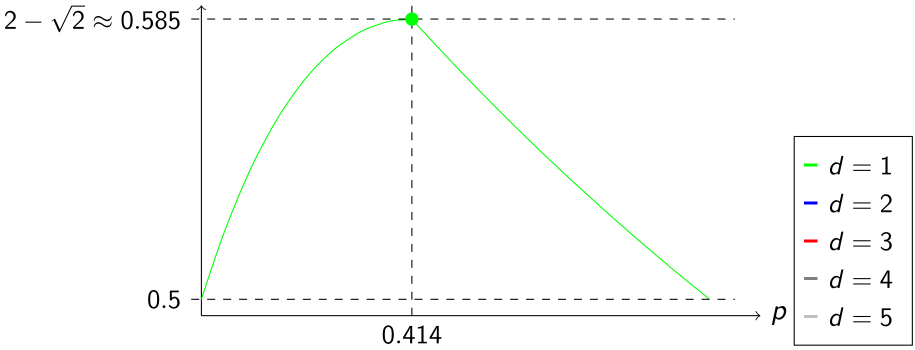
$$\left(\frac{1}{2} + \frac{p}{d+p} - \frac{p}{2d}\right) \cdot |M^*|.$$



Optimal Algorithms

The final matching returned is always at least of size

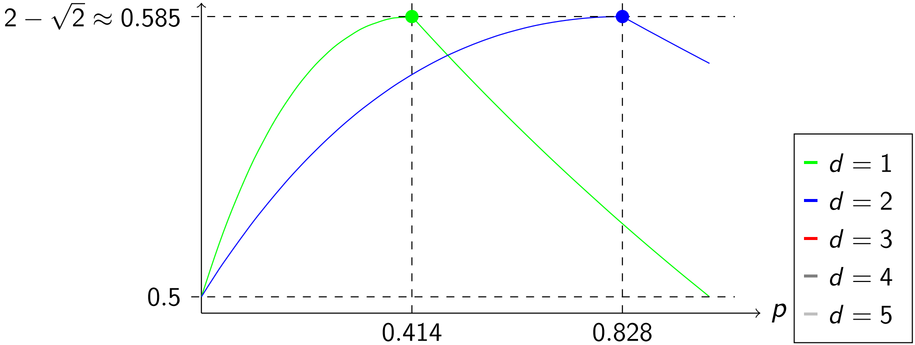
$$(\frac{1}{2} + \frac{p}{d+p} - \frac{p}{2d}) \cdot |M^*|.$$



Optimal Algorithms

The final matching returned is always at least of size

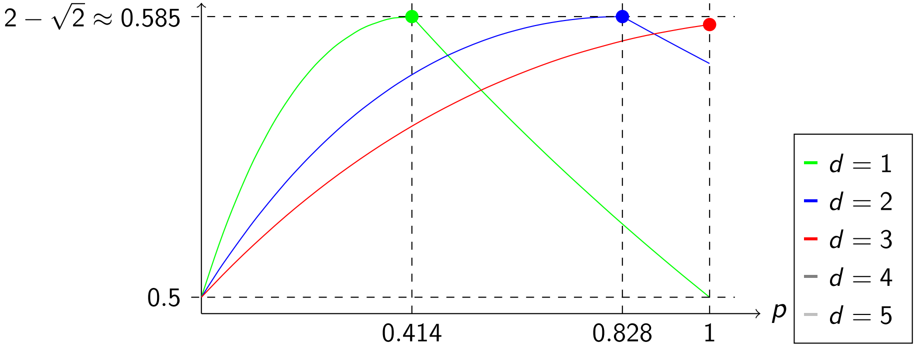
$$(\frac{1}{2} + \frac{p}{d+p} - \frac{p}{2d}) \cdot |M^*|.$$



Optimal Algorithms

The final matching returned is always at least of size

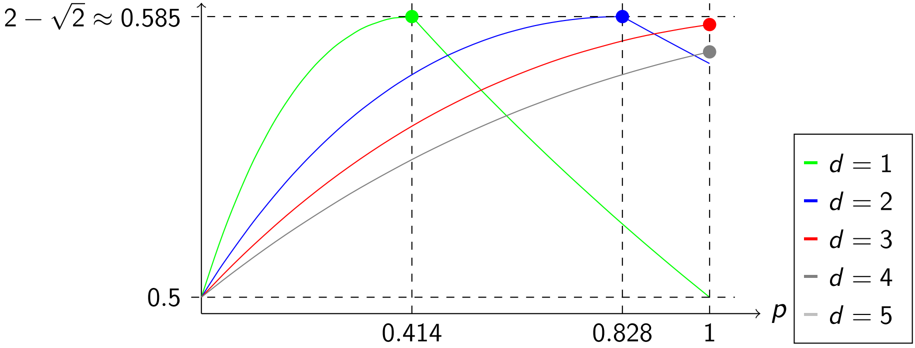
$$\left(\frac{1}{2} + \frac{p}{d+p} - \frac{p}{2d}\right) \cdot |M^*|.$$



Optimal Algorithms

The final matching returned is always at least of size

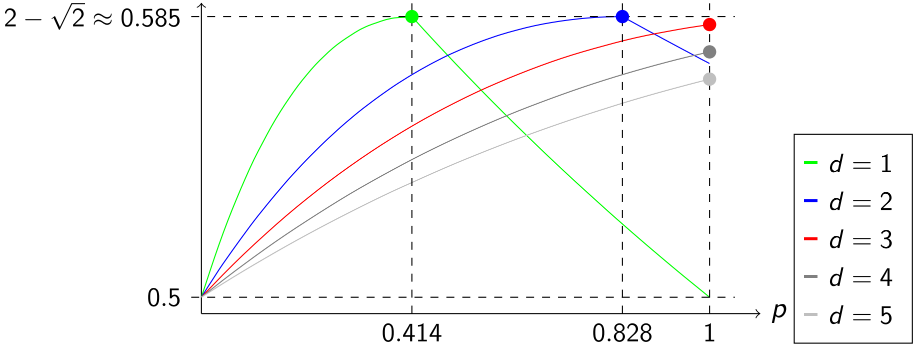
$$(\frac{1}{2} + \frac{p}{d+p} - \frac{p}{2d}) \cdot |M^*|.$$



Optimal Algorithms

The final matching returned is always at least of size

$$\left(\frac{1}{2} + \frac{p}{d+p} - \frac{p}{2d}\right) \cdot |M^*|.$$

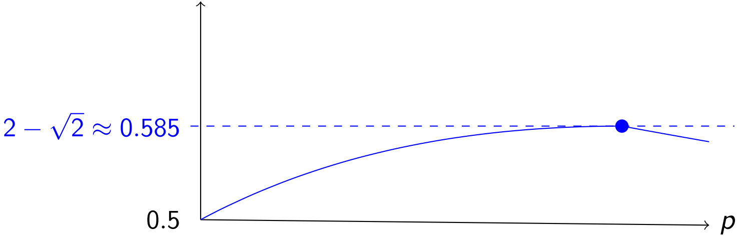


Overview

Conclusion

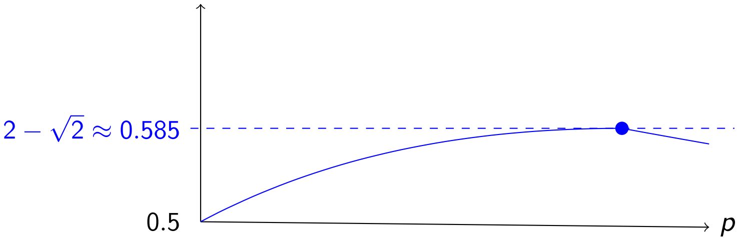
Conclusion

- Our two-pass algorithm unifies the dominant techniques used, achiving the current state-of-the-art.



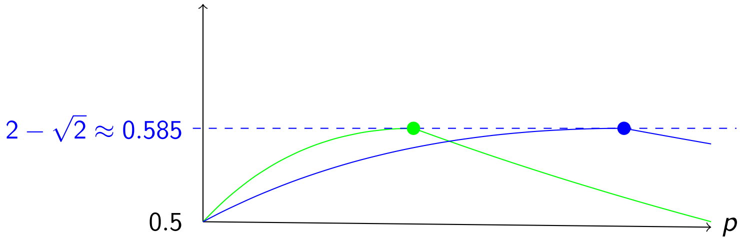
Conclusion

- Our two-pass algorithm unifies the dominant techniques used, achiving the current state-of-the-art.
- For appropriate settings of d and p, we can find Konrad's [MFCS18] and Esfandiari et al.'s [ICDMW16] algorithms.



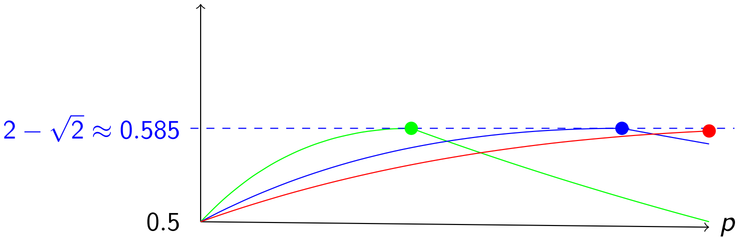
Conclusion

- Our two-pass algorithm unifies the dominant techniques used, achiving the current state-of-the-art.
- For appropriate settings of d and p, we can find Konrad's [MFCS18] and Esfandiari et al.'s [ICDMW16] algorithms.



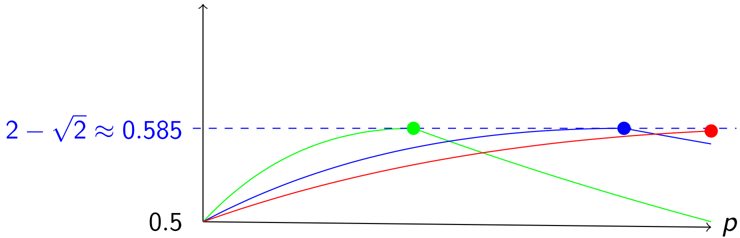
Conclusion

- Our two-pass algorithm unifies the dominant techniques used, achiving the current state-of-the-art.
- For appropriate settings of d and p, we can find Konrad's [MFCS18] and Esfandiari et al.'s [ICDMW16] algorithms.



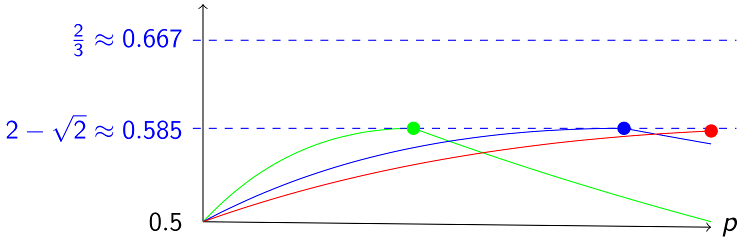
Conclusion

- Our two-pass algorithm unifies the dominant techniques used, achiving the current state-of-the-art.
- For appropriate settings of d and p, we can find Konrad's [MFCS18] and Esfandiari et al.'s [ICDMW16] algorithms.
- Our hard-instance family of graphs proves that the analysis is tight.



Conclusion

- Our two-pass algorithm unifies the dominant techniques used, achiving the current state-of-the-art.
- For appropriate settings of d and p, we can find Konrad's [MFCS18] and Esfandiari et al.'s [ICDMW16] algorithms.
- Our hard-instance family of graphs proves that the analysis is tight.
- We reduced the gap of possibility with this class of algorithms to $[0.585, 0.667]$.



Open Questions

- Can we extend other one-pass lower bounds to improve the two-pass result? I.e. Kapralov's [SODA21].
- Is there a way to do better by finding more than just a maximal matching in the first-pass?
- Can we beat a $\frac{1}{2}$-approximation in just one-pass?

Thank You