A Unifying Class of Algorithms for Semi-Streaming Bipartite Maximum Matching

Kheeran Naidu

University of Bristol

kn16063@bristol.ac.uk

Joint work with Dr Christian Konrad

Overview

1. Background
2. Our Work
   - Algorithmic
   - Impossibility
3. Discussion

Matchings

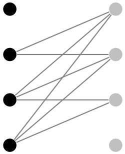
Definition

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

Matchings

Definition

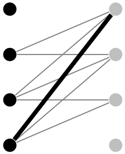
A **(bipartite) matching** is a subset of edges of a graph where every vertex has degree at most 1.



Matchings

Definition

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

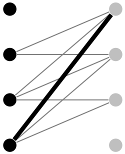


matching

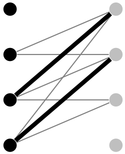
Matchings

Definition

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



matching

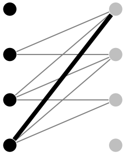


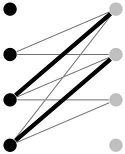
maximal matching

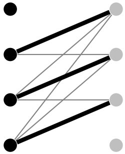
Matchings

Definition

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


matching

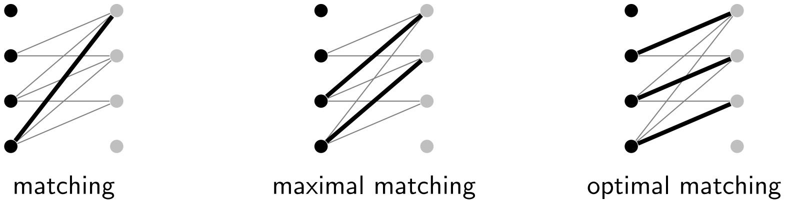

maximal matching


maximum matching

Matchings

Definition

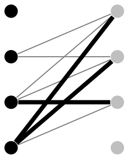
A **(bipartite) matching** is a subset of edges of a graph where every vertex has degree at most 1.



Matchings

Esfandiari et al. [ICDMW16]

A **semi-incomplete matching** is an extended matching in bipartite graphs where the vertices of one bipartition allows for degree $d > 1$.

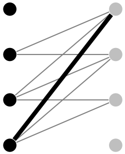


semi-incomplete matching

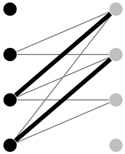
Matchings

Definition

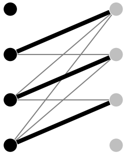
A **(bipartite) matching** is a subset of edges of a graph where every vertex has degree at most 1.



matching



maximal matching



optimal matching

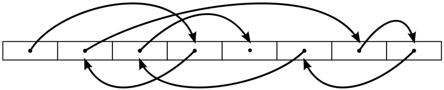
Matchings

Definition

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

Finding a maximum (optimal) matching:

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

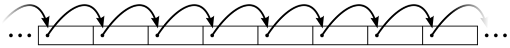


Semi-Streaming Model

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 $O(n \,\mathrm{polylog}\, n)$.

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



Main Results

Two-Pass Bipartite Maximum (Optimal) 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 Bipartite Maximum (Optimal) 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.

Main Results

Two-Pass Bipartite Maximum (Optimal) Matching

Algorithmic:

Impossibility:

Main Results

Two-Pass Bipartite Maximum (Optimal) Matching

Algorithmic:

- unify the two dominant techniques; vertex subsampling and finding a semi-incomplete matching;

Impossibility:

Main Results

Two-Pass Bipartite Maximum (Optimal) Matching

Algorithmic:

- unify the two dominant techniques; vertex subsampling and finding a semi-incomplete matching;
- present a wider set of algorithms which contains the most recent state-of-the-art results;

Impossibility:

Main Results

Two-Pass Bipartite Maximum (Optimal) Matching

Algorithmic:

- unify the two dominant techniques; vertex subsampling and finding a semi-incomplete matching;
- present a wider set of algorithms which contains the most recent state-of-the-art results;
- find a novel meta algorithm that exactly achieves the current state-of-the-art $2 - \sqrt{2} \approx \frac{1}{2} + 0.085$-approximation.

Impossibility:

Main Results

Two-Pass Bipartite Maximum (Optimal) Matching

Algorithmic:

- unify the two dominant techniques; vertex subsampling and finding a semi-incomplete matching;
- present a wider set of algorithms which contains the most recent state-of-the-art results;
- find a novel meta algorithm that exactly achieves the current state-of-the-art $2 - \sqrt{2} \approx \frac{1}{2} + 0.085$-approximation.

Impossibility:

- In this class of algorithms, *(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 the semi-streaming model.

Related Work

	Algorithmic	Impossibility
one-pass	$\frac{1}{2}$ [ICALP04]	$\frac{1}{2} + 0.167$ [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

1 Background

2 Our Work
- Algorithmic
- Impossibility

3 Discussion

Overview

1. Background

2. Our Work
   - Algorithmic
   - Impossibility

3. Discussion

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 an edge not in the matching, and alternates along the path between edges in and out of the matching.

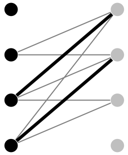
Algorithmic Approach

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 an edge not in the matching, and alternates along the path between edges in and out of the matching.



$|M_0| = 2$

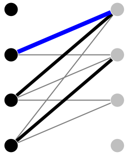
Algorithmic Approach

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 an edge not in the matching, and alternates along the path between edges in and out of the matching.



$|M_0| = 2$

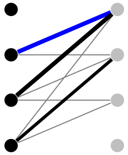
Algorithmic Approach

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 an edge not in the matching, and alternates along the path between edges in and out of the matching.



$|M_0| = 2$

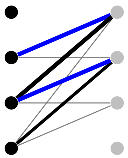
Algorithmic Approach

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 an edge not in the matching, and alternates along the path between edges in and out of the matching.



$|M_0| = 2$

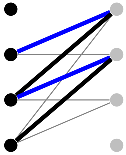
Algorithmic Approach

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 an edge not in the matching, and alternates along the path between edges in and out of the matching.



$|M_0| = 2$

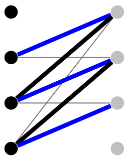
Algorithmic Approach

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 an edge not in the matching, and alternates along the path between edges in and out of the matching.



$|M_0| = 2$

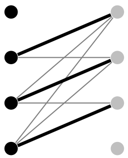
Algorithmic Approach

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 an edge not in the matching, and alternates along the path between edges in and out of the matching.

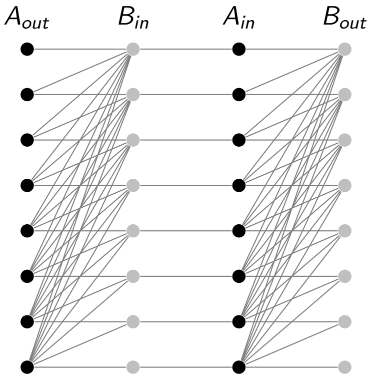


$|M_1| = 3$

Class of Algorithms

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

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

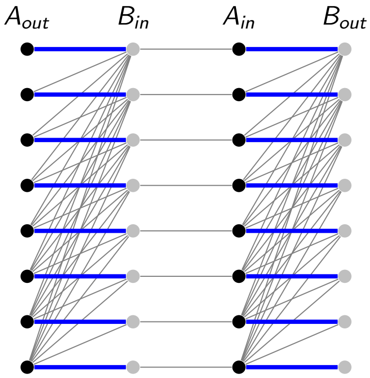


Class of Algorithms

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

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

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



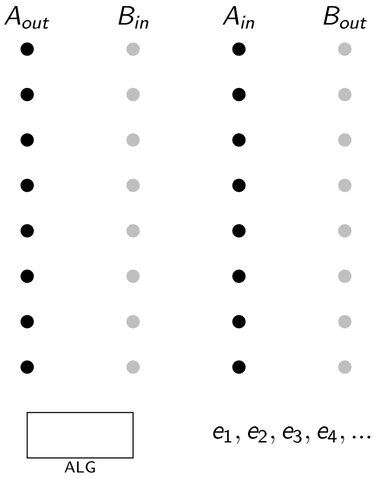
Class of Algorithms

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

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

First pass:

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



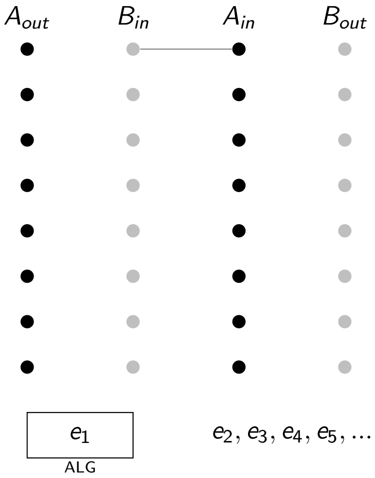
Class of Algorithms

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

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

First pass:

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



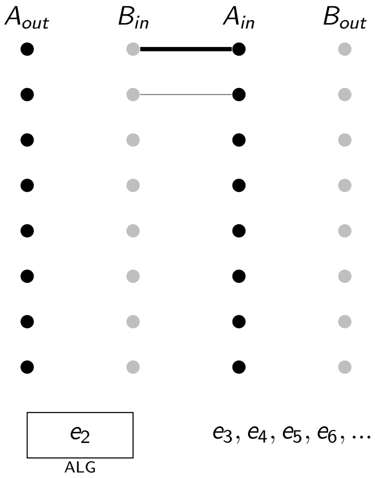
Class of Algorithms

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

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

First pass:

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



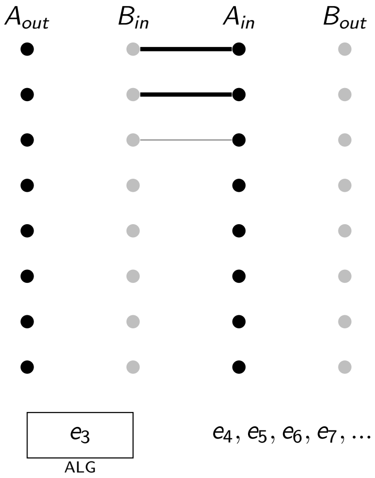
Class of Algorithms

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

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

First pass:

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



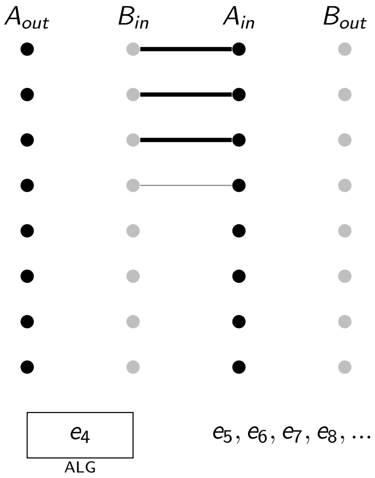
Class of Algorithms

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

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

First pass:

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

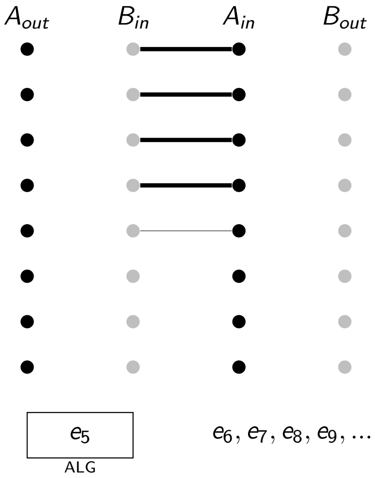


Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

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

First pass:

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



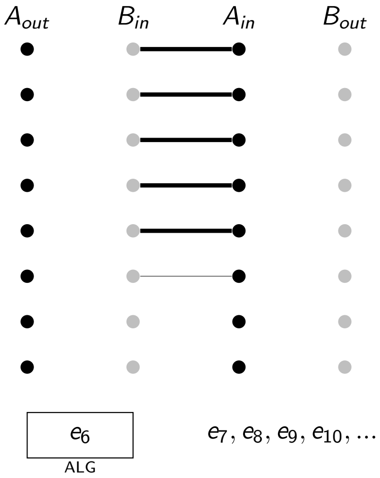
Class of Algorithms

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

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

First pass:

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



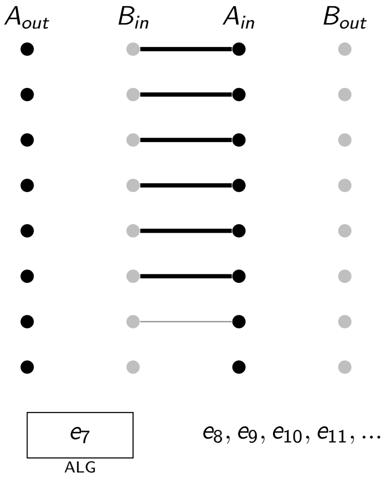
Class of Algorithms

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

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

First pass:

- $M \leftarrow \text{Greedy}(\pi_G)$



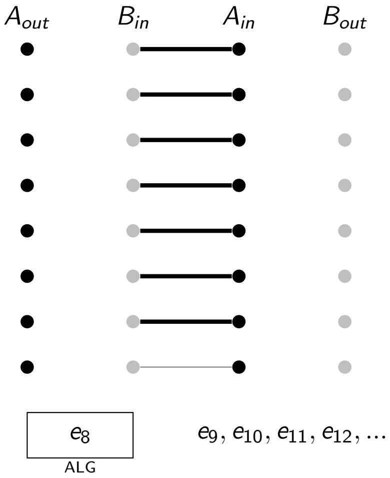
Class of Algorithms

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

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

First pass:

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



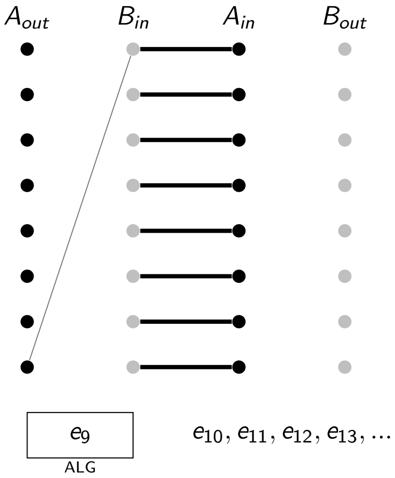
Class of Algorithms

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

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

First pass:

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



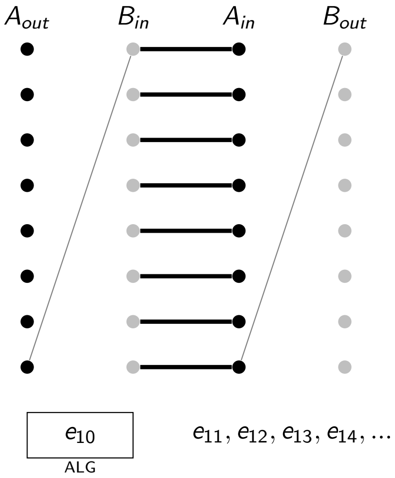
Class of Algorithms

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

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

First pass:

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



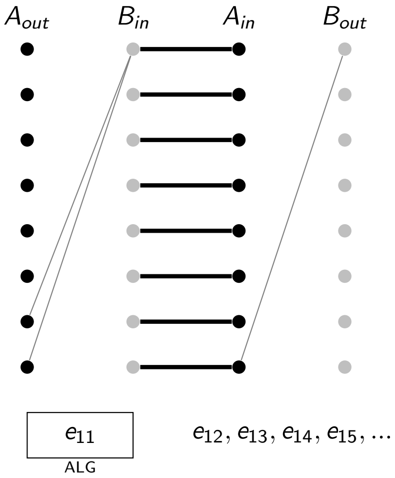
Class of Algorithms

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

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

First pass:

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



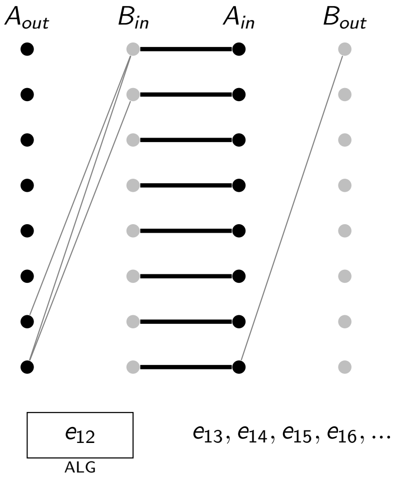
Class of Algorithms

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

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

First pass:

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



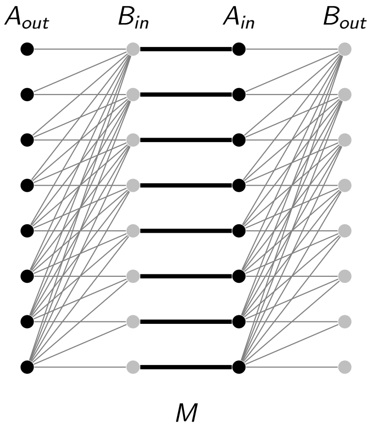
Class of Algorithms

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

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

First pass:

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

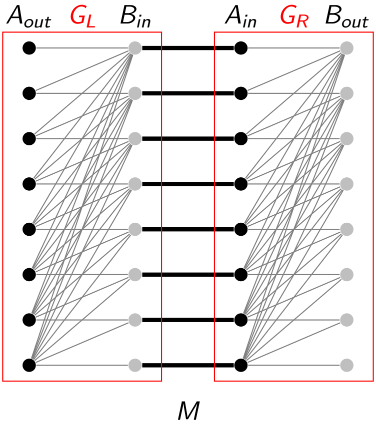


Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

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

First pass:

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



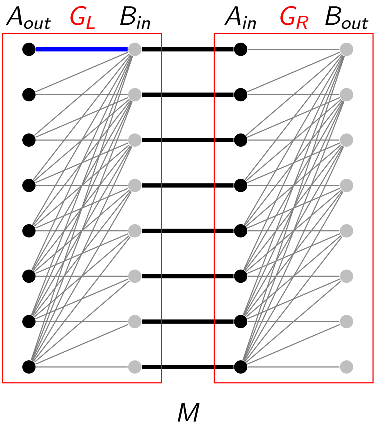
Class of Algorithms

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

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

First pass:

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



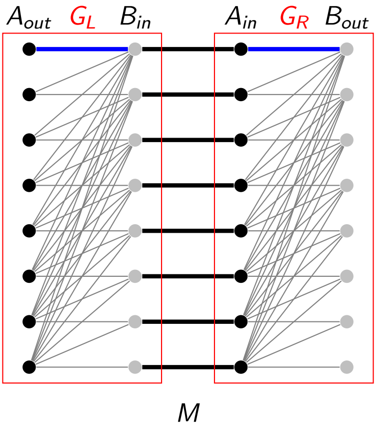
Class of Algorithms

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

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

First pass:

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

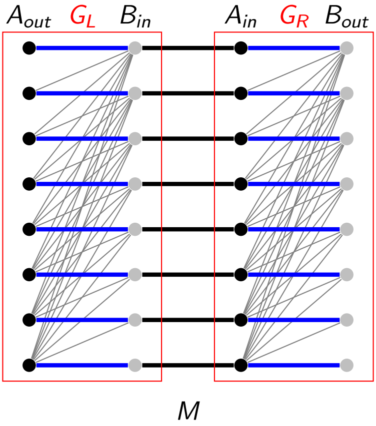


Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

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

First pass:

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



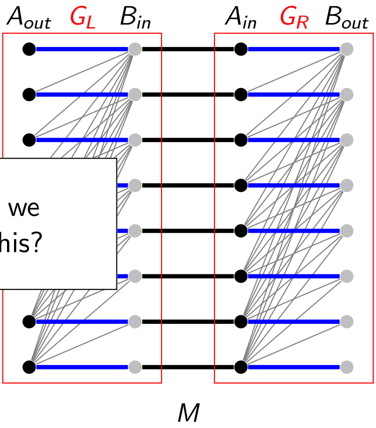
Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

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

First pass:

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

How can we achieve this?



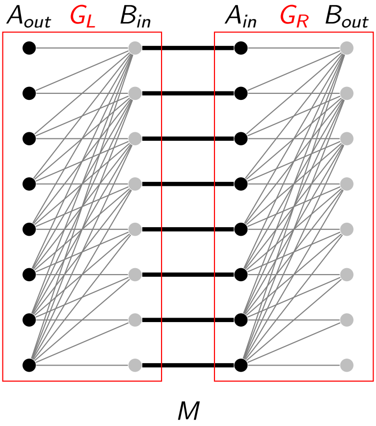
Class of Algorithms

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

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

First pass:

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

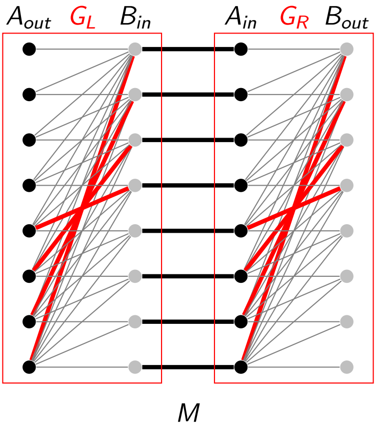


Let $G = (A, B, E)$ be a hard-instance (worst-case) graph.

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

First pass:

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



Class of Algorithms

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

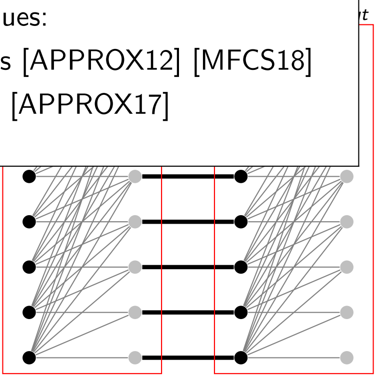
Let π_G
edges
order.

Using the two dominant techniques:

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

First pass:

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



Let $G = (A, B, E)$ be a hard-instance

(worst

Let π

edges

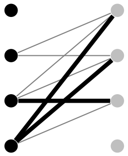
order.

First

- M

Using the two dominant techniques:

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



semi-incomplete matching

M

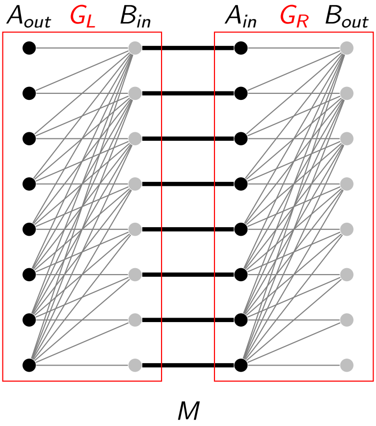
Class of Algorithms

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

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

First pass:

- $M \leftarrow \text{GREEDY}(\pi_G)$



Class of Algorithms

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

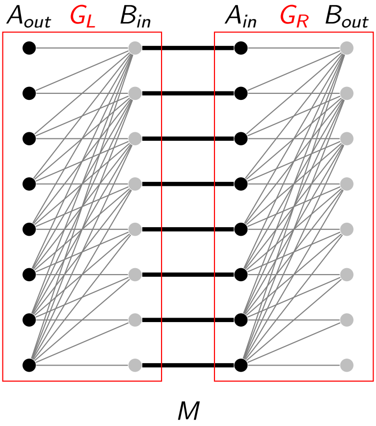
Let $\pi_G = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

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

Second pass:

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



Class of Algorithms

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

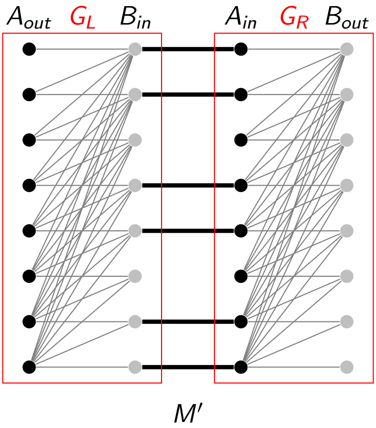
Let $\pi_G = e_1, e_2, \dots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

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

Second pass:

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



$d = 3,\ p = 0.67$

Class of Algorithms

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

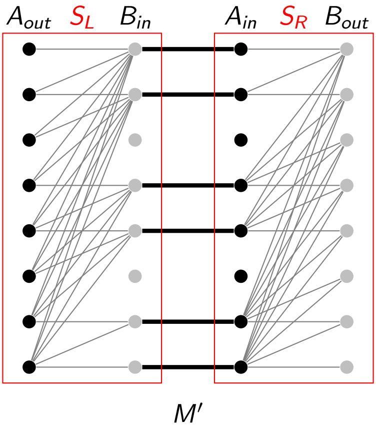
Let $\pi_G = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

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

Second pass:

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



$d = 3,\ p = 0.67$

Class of Algorithms

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

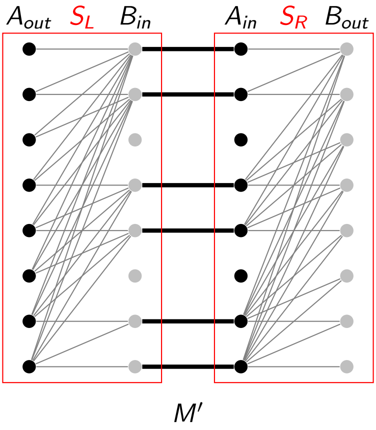
Let $\pi_G = e_1, e_2, \dots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

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

Second pass:

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



$d = 3,\ p = 0.67$

Class of Algorithms

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

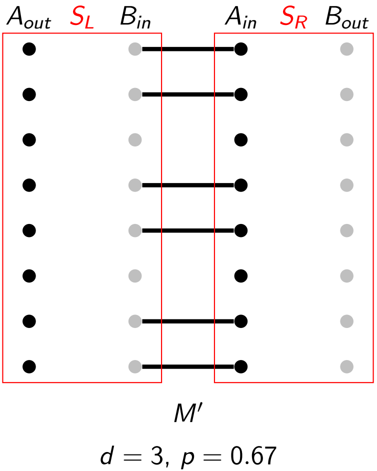
Let $\pi_G = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

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

Second pass:

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



Class of Algorithms

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

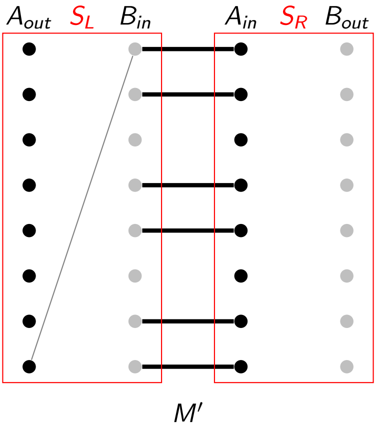
Let $\pi_G = e_1, e_2, \dots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

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

Second pass:

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



$d = 3, \ p = 0.67$

Class of Algorithms

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

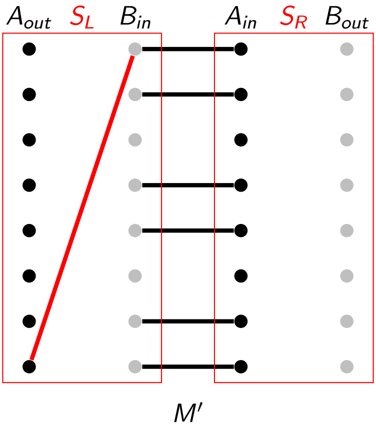
Let $\pi_G = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

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

Second pass:

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



$d = 3, \ p = 0.67$

Class of Algorithms

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

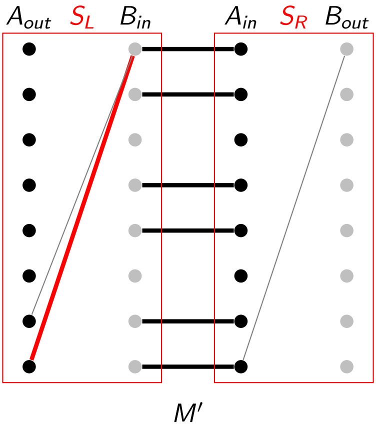
Let $\pi_G = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

- $M \leftarrow \text{GREEDY}(\pi_G)$

Second pass:

- subsample $M' \subseteq M$ with prob. p
- $S_L \leftarrow E \cap A_{out} \times B(M')$
- $S_R \leftarrow E \cap A(M') \times B_{out}$
- $M'_L \leftarrow \text{GREEDY}_d(S_L \cap \pi_G)$
- $M'_R \leftarrow \text{GREEDY}_d(S_R \cap \pi_G)$



$d = 3,\ p = 0.67$

Class of Algorithms

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

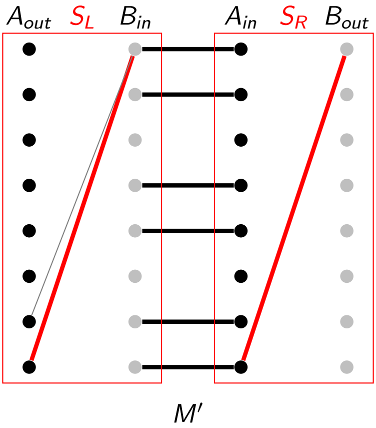
Let $\pi_G = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

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

Second pass:

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



$d = 3,\ p = 0.67$

Class of Algorithms

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

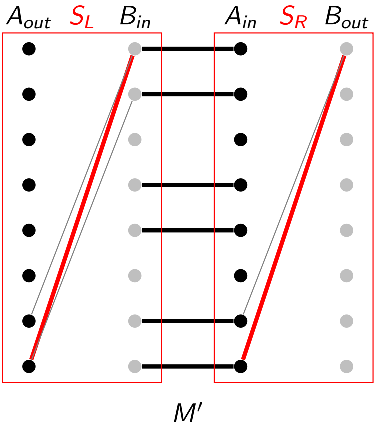
Let $\pi_G = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

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

Second pass:

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



$d = 3, \ p = 0.67$

Class of Algorithms

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

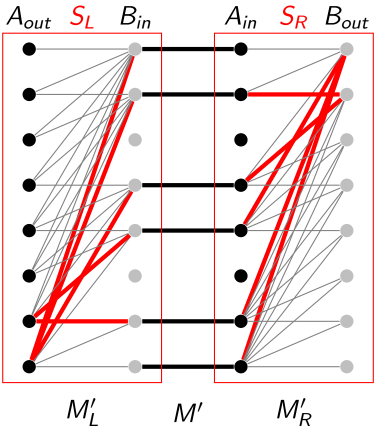
Let $\pi_G = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

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

Second pass:

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



$d = 3, \; p = 0.67$

Class of Algorithms

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

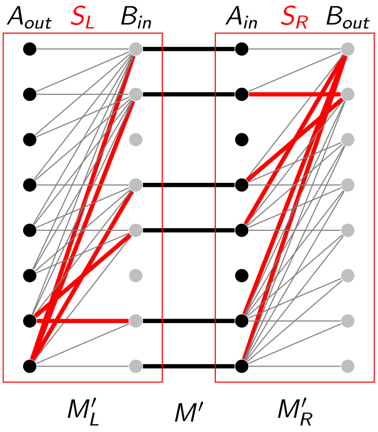
Let $\pi_G = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

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

Second pass:

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



$d = 3,\ p = 0.67$

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

Class of Algorithms

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

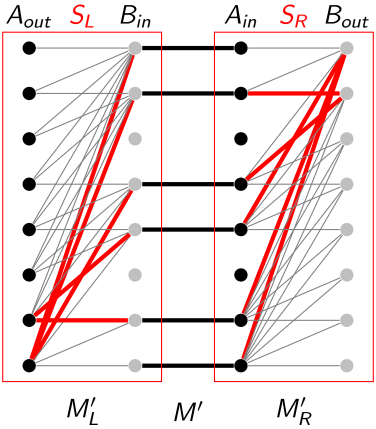
Let $\pi_G = e_1, e_2, \ldots$ be a stream of its edges in adversarial (worst-case) order.

First pass:

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

Second pass:

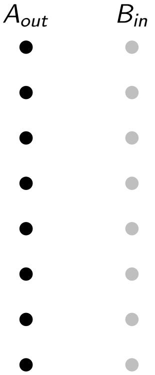
- subsample $M' \subseteq M$ with prob. p
- $S_L \leftarrow E \cap A_{out} \times B(M')$
- $S_R \leftarrow E \cap A(M') \times B_{out}$
- $M'_L \leftarrow \textsc{Greedy}_d(S_L \cap \pi_G)$
- $M'_R \leftarrow \textsc{Greedy}_d(S_R \cap \pi_G)$



$d = 3,\ p = 0.67$

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

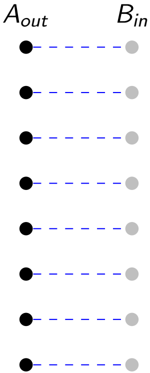
Main Theorem (Proof Outline)



Main Theorem (Proof Outline)

Setup:

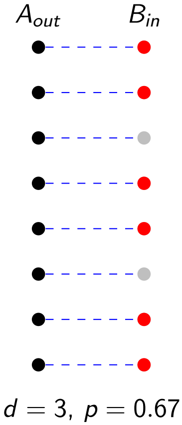
- any bipartite graph $G = (A, B, E)$ with a maximum (optimal) matching M^*;



Main Theorem (Proof Outline)

Setup:

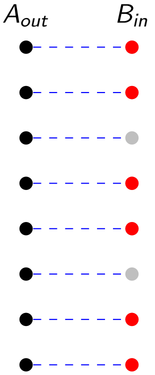
- any bipartite graph $G = (A, B, E)$ with a maximum (optimal) matching M^*;
- subsample $B' \subseteq B$ with prob. p;



Main Theorem (Proof Outline)

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum (optimal) matching M^*;
- subsample $B' \subseteq B$ with prob. p;
- $M \leftarrow \textsc{Greedy}_d$.



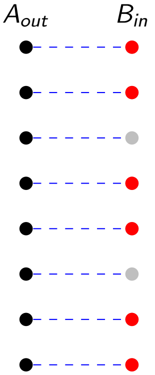
$d = 3,\ p = 0.67$

Main Theorem (Proof Outline)

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum (optimal) matching M^*;
- subsample $B' \subseteq B$ with prob. p;
- $M \leftarrow \textsc{Greedy}_d$.

Proof:



$d = 3,\ p = 0.67$

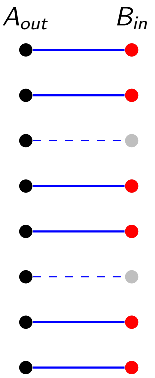
Main Theorem (Proof Outline)

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum (optimal) matching M^*;
- subsample $B' \subseteq B$ with prob. p;
- $M \leftarrow \textsc{Greedy}_d$.

Proof:

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



$d = 3,\ p = 0.67$

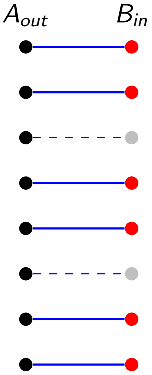
Main Theorem (Proof Outline)

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum (optimal) matching M^*;
- subsample $B' \subseteq B$ with prob. p;
- $M \leftarrow \textsc{Greedy}_d$.

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?



$d = 3,\ p = 0.67$

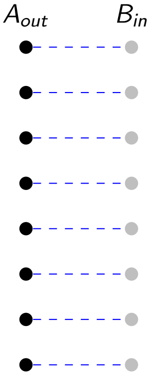
Main Theorem (Proof Outline)

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum (optimal) matching M^*;
- subsample $B' \subseteq B$ with prob. p;
- $M \leftarrow \textsc{Greedy}_d$.

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?



$d = 3$, $p = 0.67$

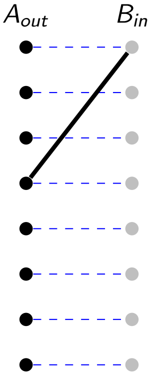
Main Theorem (Proof Outline)

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum (optimal) matching M^*;
- subsample $B' \subseteq B$ with prob. p;
- $M \leftarrow \textsc{Greedy}_d$.

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?



$d = 3,\ p = 0.67$

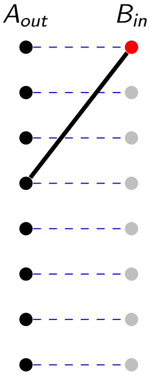
Main Theorem (Proof Outline)

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum (optimal) matching M^*;
- subsample $B' \subseteq B$ with prob. p;
- $M \leftarrow \textsc{Greedy}_d$.

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?



$d = 3,\ p = 0.67$

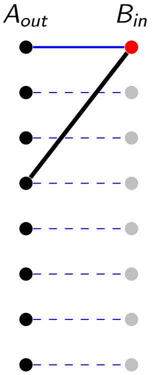
Main Theorem (Proof Outline)

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum (optimal) matching M^*;
- subsample $B' \subseteq B$ with prob. p;
- $M \leftarrow \textsc{Greedy}_d$.

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?



$d = 3,\ p = 0.67$

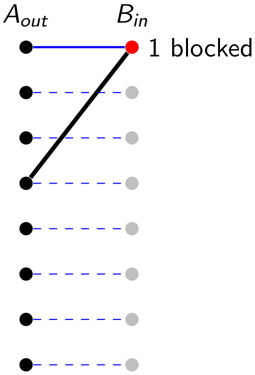
Main Theorem (Proof Outline)

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum (optimal) matching M^*;
- subsample $B' \subseteq B$ with prob. p;
- $M \leftarrow \textsc{Greedy}_d$.

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?



$d = 3,\ p = 0.67$

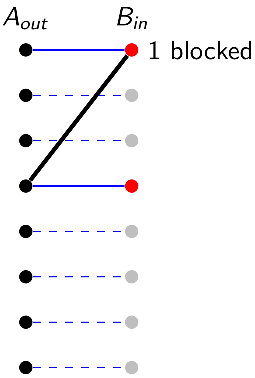
Main Theorem (Proof Outline)

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum (optimal) matching M^*;
- subsample $B' \subseteq B$ with prob. p;
- $M \leftarrow \text{GREEDY}_d$.

Proof:

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



$d = 3$, $p = 0.67$

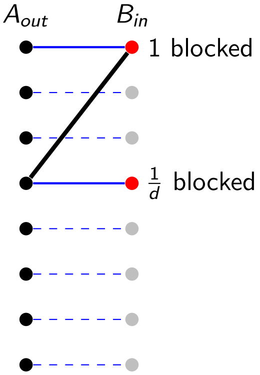
Main Theorem (Proof Outline)

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum (optimal) matching M^*;
- subsample $B' \subseteq B$ with prob. p;
- $M \leftarrow \textsc{Greedy}_d$.

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?



$d = 3,\ p = 0.67$

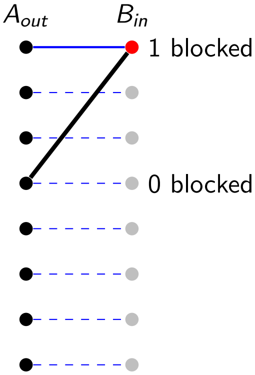
Main Theorem (Proof Outline)

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum (optimal) matching M^*;
- subsample $B' \subseteq B$ with prob. p;
- $M \leftarrow \textsc{Greedy}_d$.

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?



$d = 3,\ p = 0.67$

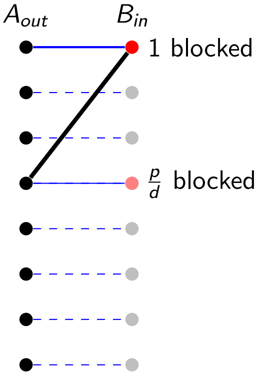
Main Theorem (Proof Outline)

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum (optimal) matching M^*;
- subsample $B' \subseteq B$ with prob. p;
- $M \leftarrow \textsc{Greedy}_d$.

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?



$d = 3$, $p = 0.67$

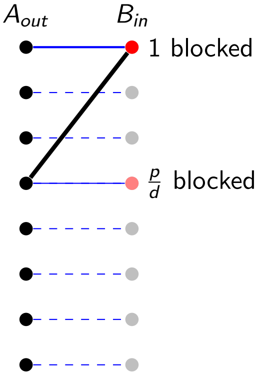
Main Theorem (Proof Outline)

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum (optimal) matching M^*;
- subsample $B' \subseteq B$ with prob. p;
- $M \leftarrow \textsc{Greedy}_d$.

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?



$d = 3,\ p = 0.67$

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

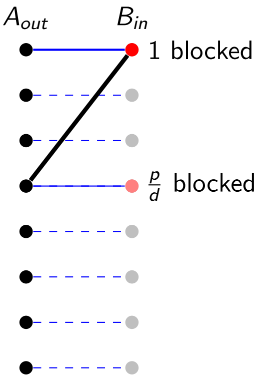
Main Theorem (Proof Outline)

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum (optimal) matching M^*;
- subsample $B' \subseteq B$ with prob. p;
- $M \leftarrow \textsc{Greedy}_d$.

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?



$d = 3,\ p = 0.67$

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

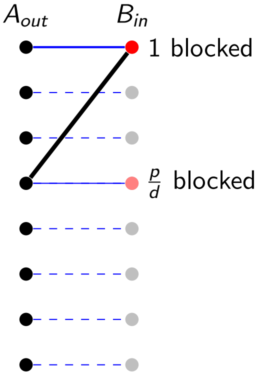
Main Theorem (Proof Outline)

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum (optimal) matching M^*;
- subsample $B' \subseteq B$ with prob. p;
- $M \leftarrow \textsc{Greedy}_d$.

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?



$d = 3,\ p = 0.67$

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

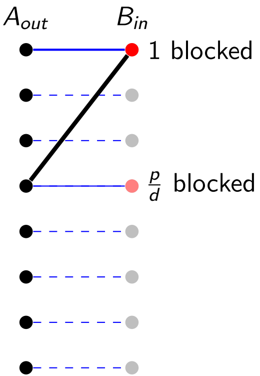
Main Theorem (Proof Outline)

Setup:

- any bipartite graph $G = (A, B, E)$ with a maximum (optimal) matching M^*;
- subsample $B' \subseteq B$ with prob. p;
- $M \leftarrow \textsc{Greedy}_d$.

Proof:

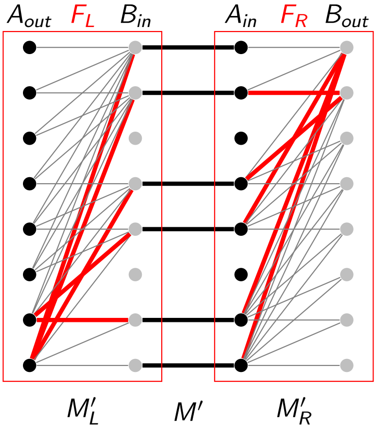
- $M^*_{B'} \leftarrow \{ab \in M^* : b \in B'\}$;
- For every edge that is added by $\textsc{Greedy}_d$, how many edges in $M^*_{B'}$ are blocked?
- Formalised using Wald's Equation.



$d = 3,\ p = 0.67$

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

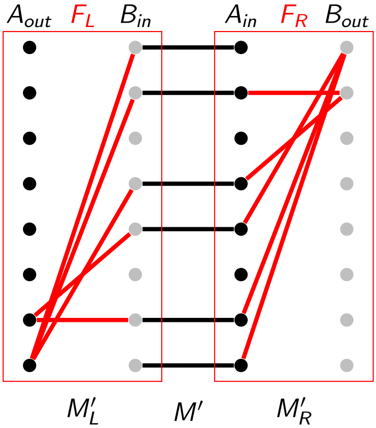
Optimal Algorithms I



$d = 3,\ p = 0.67$

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

Optimal Algorithms I



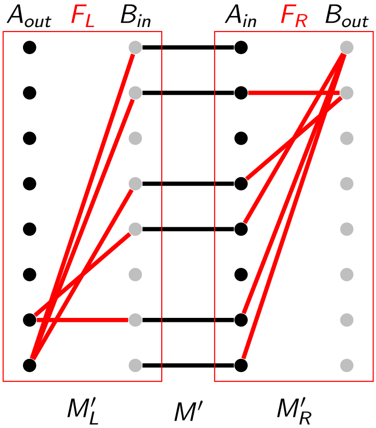
$d = 3, \; p = 0.67$

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

Optimal Algorithms I

Analysis:

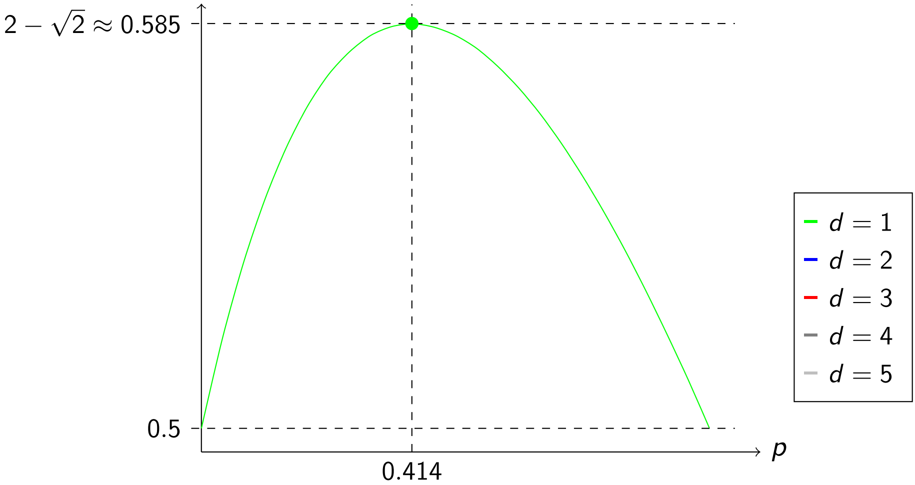
We find at least $(\frac{1}{2} + \frac{p}{d+p} - \frac{p}{2d}) \cdot |M^*|$ edges in our final matching.



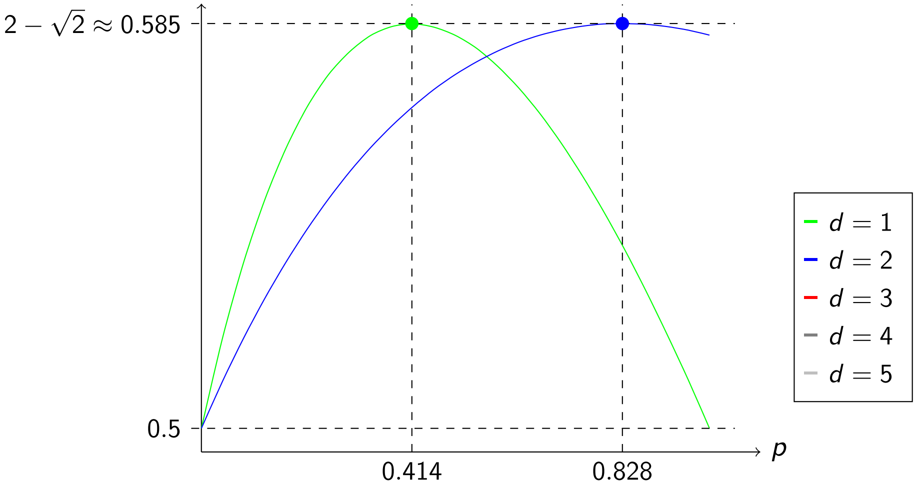
$d = 3,\ p = 0.67$

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

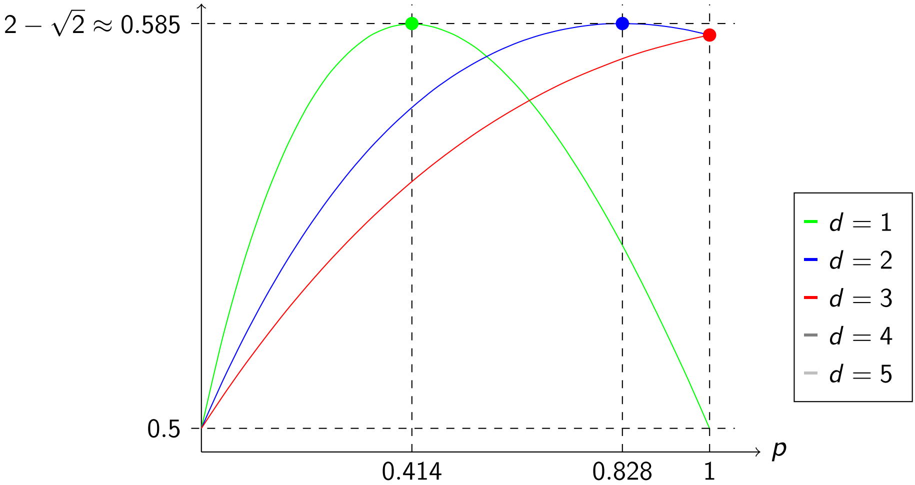
Optimal Algorithms II



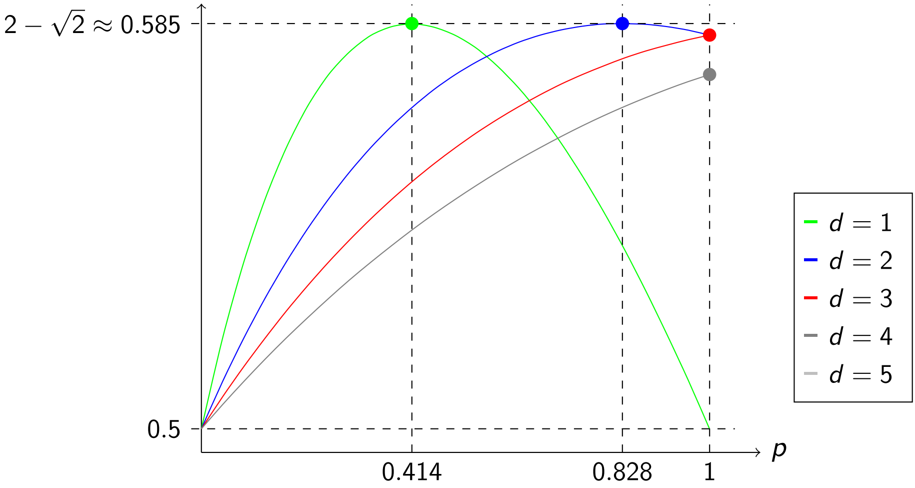
Optimal Algorithms II



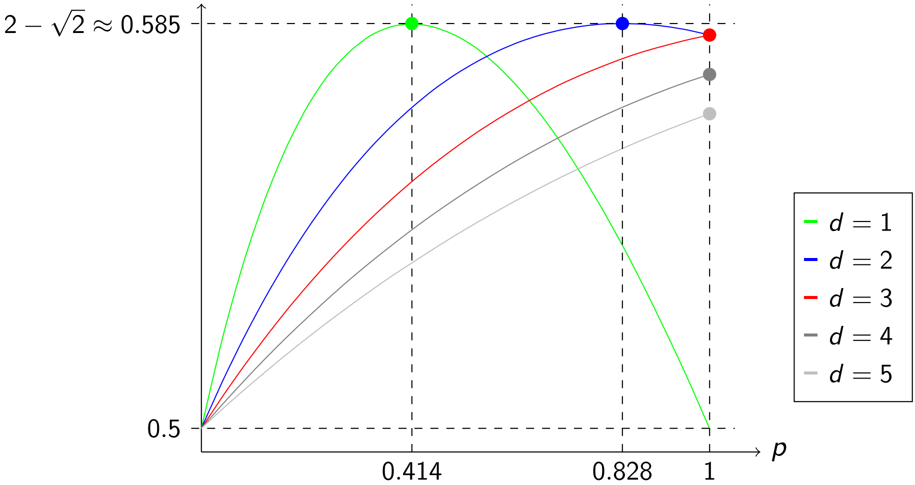
Optimal Algorithms II



Optimal Algorithms II



Optimal Algorithms II



Overview

1. Background
2. Our Work
   - Algorithmic
   - Impossibility
3. Discussion

Two-Party Communication Setup

$$\boxed{\text{Alice}} \longrightarrow \boxed{\text{Bob}} \longrightarrow \boxed{M^*}$$

Proving Impossibility Results

Two-Party Communication Setup

$$\boxed{\text{Alice}} \longrightarrow \boxed{\text{Bob}} \longrightarrow \boxed{M^*}$$

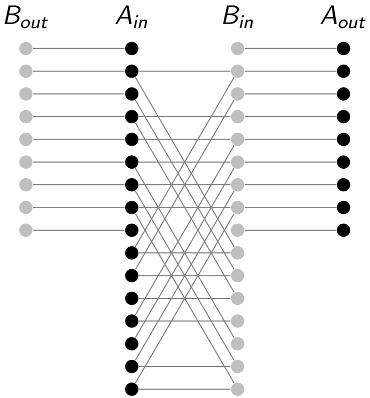
Goal:

- to bound the size of the message from Alice to Bob needed to output a large matching.

One-Pass Impossibility Proof [SODA12]

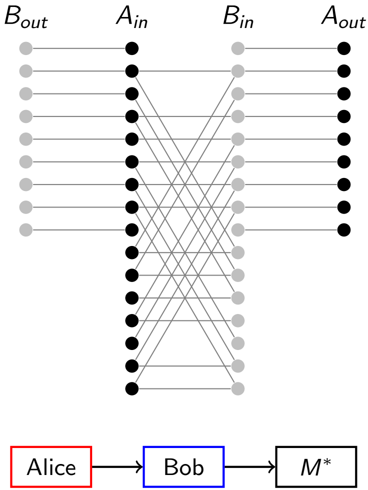
Outline:

1. A family of graphs constructed from very dense Rusza-Szemeredi (RS) graphs.



Outline:

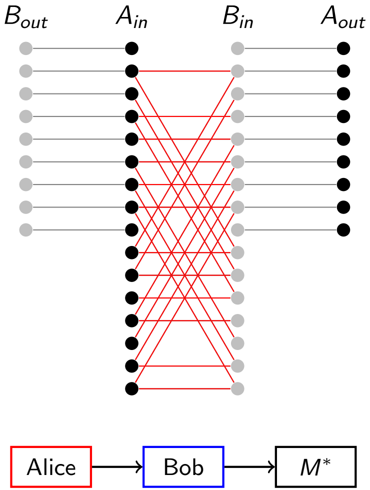
1. A family of graphs constructed from very dense Rusza-Szemeredi (RS) graphs.



One-Pass Impossibility Proof [SODA12]

Outline:

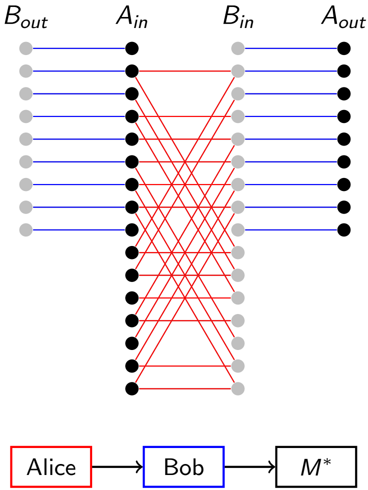
1. A family of graphs constructed from very dense Rusza-Szemeredi (RS) graphs.



One-Pass Impossibility Proof [SODA12]

Outline:

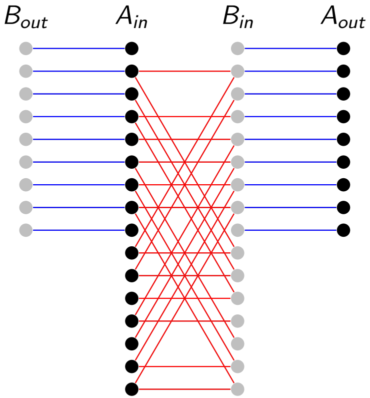
1. A family of graphs constructed from very dense Rusza-Szemeredi (RS) graphs.



One-Pass Impossibility Proof [SODA12]

Outline:

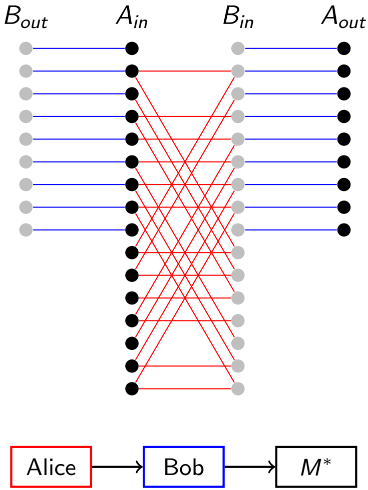
1. A family of graphs constructed from very dense Rusza-Szemeredi (RS) graphs.
2. Prove that a $(\frac{2}{3} + \epsilon)$-approx requires $N^{1+\Omega(\frac{1}{\log \log N})} \supset O(N \text{ polylog } N)$ space.



Two-Pass Impossibility Proof

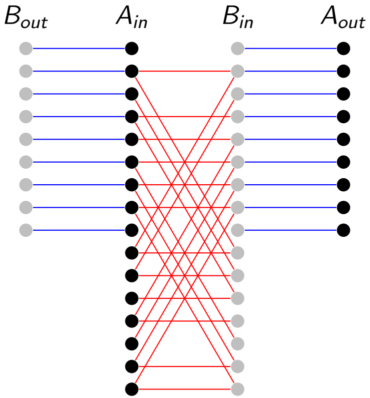
Class of algorithms:

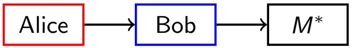
1. finds a maximal matching M (first-pass);
2. increases the size of the matching M (second-pass).



Two-Pass Impossibility Proof

Goal: Give both Alice and Bob knowledge of a maximal matching without affecting the difficulty of the problem.



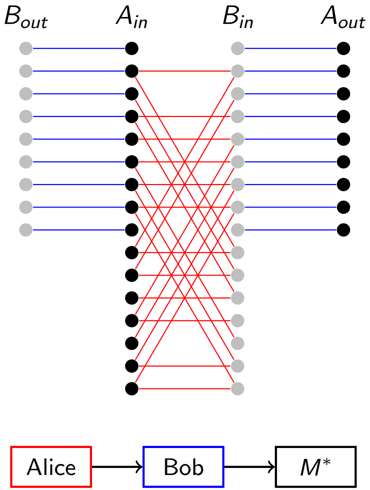


Two-Pass Impossibility Proof

Goal: Give both Alice and Bob knowledge of a maximal matching without affecting the difficulty of the problem.

Outline:

- We extend [SODA12]'s construction a family of RS graphs.

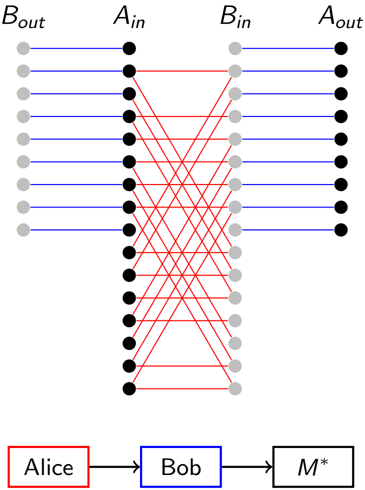


Two-Pass Impossibility Proof

Goal: Give both Alice and Bob knowledge of a maximal matching without affecting the difficulty of the problem.

Outline:

- We extend [SODA12]'s construction a family of RS graphs.
- Do dense RS graphs contain perfect matchings?

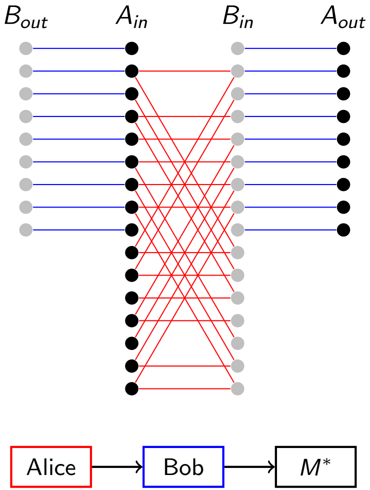


Two-Pass Impossibility Proof

Goal: Give both Alice and Bob knowledge of a maximal matching without affecting the difficulty of the problem.

Outline:

- We extend [SODA12]'s construction a family of RS graphs.
- Do dense RS graphs contain perfect matchings?
- Find a near-perfect matching of size $N - \epsilon' N$

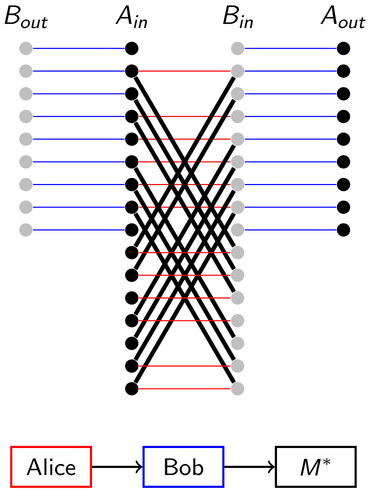


Two-Pass Impossibility Proof

Goal: Give both Alice and Bob knowledge of a maximal matching without affecting the difficulty of the problem.

Outline:

- We extend [SODA12]'s construction a family of RS graphs.
- Do dense RS graphs contain perfect matchings?
- Find a near-perfect matching of size $N - \epsilon' N$

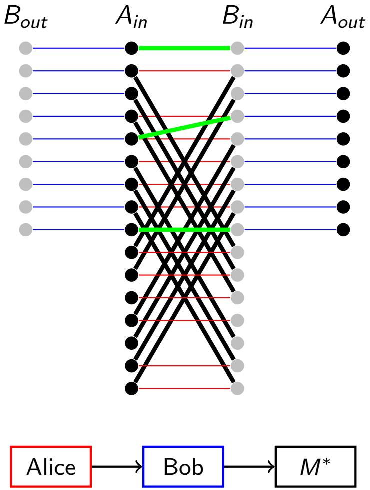


Two-Pass Impossibility Proof

Goal: Give both Alice and Bob knowledge of a maximal matching without affecting the difficulty of the problem.

Outline:

- We extend [SODA12]'s construction a family of RS graphs.
- Do dense RS graphs contain perfect matchings?
- Find a near-perfect matching of size $N - \epsilon' N$

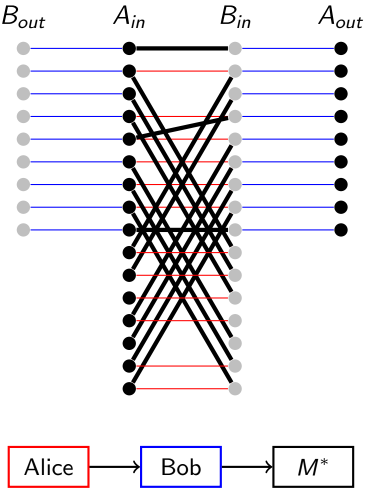


Two-Pass Impossibility Proof

Goal: Give both Alice and Bob knowledge of a maximal matching without affecting the difficulty of the problem.

Outline:

- We extend [SODA12]'s construction a family of RS graphs.
- Do dense RS graphs contain perfect matchings?
- Find a near-perfect matching of size $N - \epsilon' N$
- $(\frac{2}{3} + \epsilon)$-approx requires space $N^{1+\Omega(\frac{1}{\log \log N})} \supset O(N \operatorname{polylog} N)$.



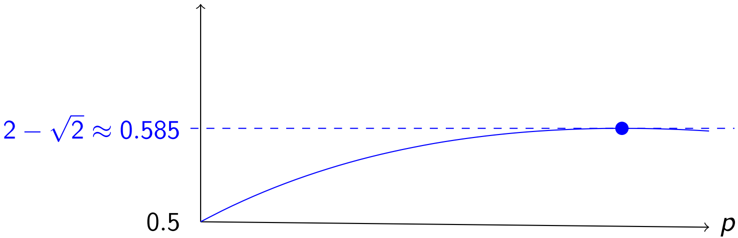
Overview

1. Background
2. Our Work
   - Algorithmic
   - Impossibility
3. Discussion

Conclusion

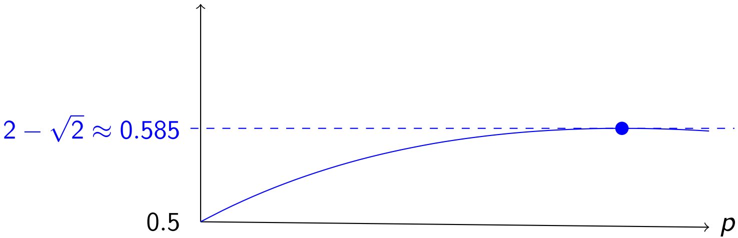
Conclusion

- Our set algorithms unifies the dominant techniques used to tackle the two-pass bipartite maximum matching problem, achieving the current state-of-the-art.



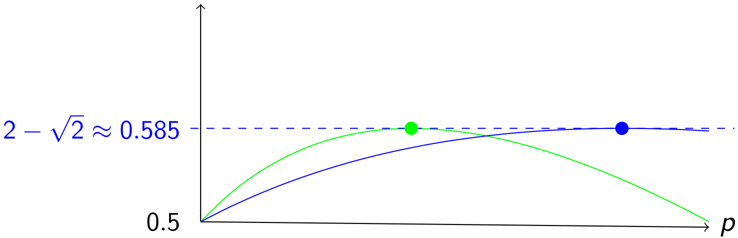
Conclusion

- Our set algorithms unifies the dominant techniques used to tackle the two-pass bipartite maximum matching problem, achieving 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.



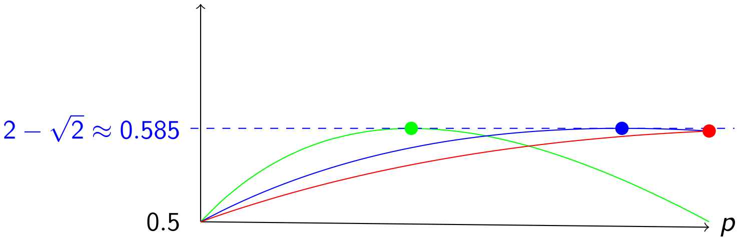
Conclusion

- Our set algorithms unifies the dominant techniques used to tackle the two-pass bipartite maximum matching problem, achieving 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.



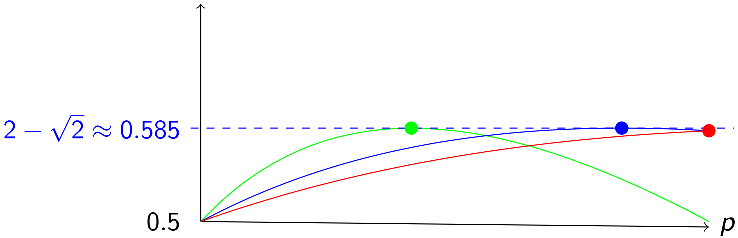
Conclusion

- Our set algorithms unifies the dominant techniques used to tackle the two-pass bipartite maximum matching problem, achieving 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.



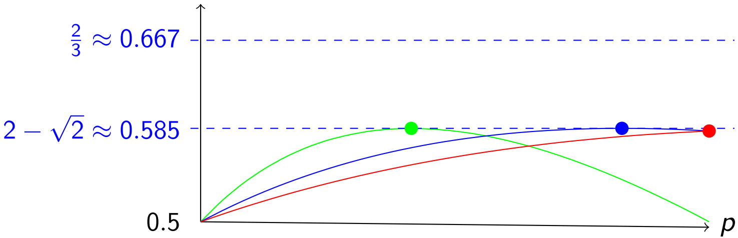
Conclusion

- Our set algorithms unifies the dominant techniques used to tackle the two-pass bipartite maximum matching problem, achieving 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 graph proves that our analysis is tight.



Conclusion

- Our set algorithms unifies the dominant techniques used to tackle the two-pass bipartite maximum matching problem, achieving 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 graph proves that our 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 impossibility results to improve the two-pass bound? 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