# Robust Communication Complexity of Approximate Maximum Matching

Adithya Diddapur[1], Pavel Dvorák[2], Christian Konrad[1]

[1]University of Bristol, [2]Charles University

# The Communication Complexity Setting

# The Communication Complexity Setting

- Two-Party One-Way Communication Complexity.

# The Communication Complexity Setting

- Two-Party One-Way Communication Complexity.

Alice

# The Communication Complexity Setting

- Two-Party One-Way Communication Complexity.

# The Communication Complexity Setting

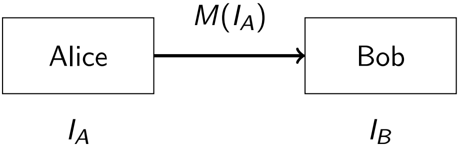
- Two-Party One-Way Communication Complexity.

# The Communication Complexity Setting

- Two-Party One-Way Communication Complexity.

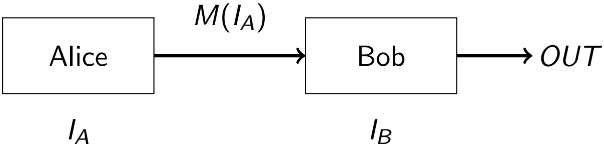
# The Communication Complexity Setting

- Two-Party One-Way Communication Complexity.



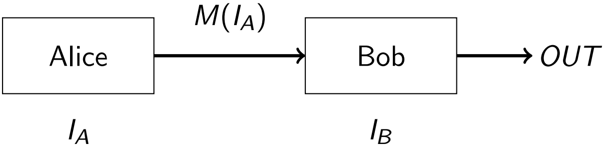
# The Communication Complexity Setting

- Two-Party One-Way Communication Complexity.



# The Communication Complexity Setting

- Two-Party One-Way Communication Complexity.




- **Aim:** Compute some function $f(I_A, I_B)$ with a message which is as small as possible.

# Approximate Maximum Matching

# Approximate Maximum Matching

- A matching is a set of vertex disjoint edges in a graph.

# Approximate Maximum Matching

- A matching is a set of vertex disjoint edges in a graph.

### Problem

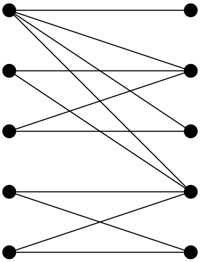
Compute a matching which is as large as possible (compared to a maximum matching).

# Approximate Maximum Matching

- A matching is a set of vertex disjoint edges in a graph.

### Problem

Compute a matching which is as large as possible (compared to a maximum matching).

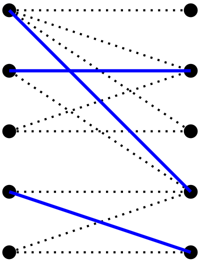


# Approximate Maximum Matching

- A matching is a set of vertex disjoint edges in a graph.

### Problem

Compute a matching which is as large as possible (compared to a maximum matching).

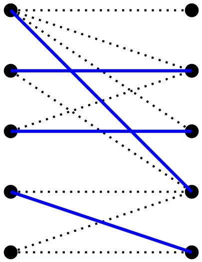


# Approximate Maximum Matching

- A matching is a set of vertex disjoint edges in a graph.

## Problem

Compute a matching which is as large as possible (compared to a maximum matching).

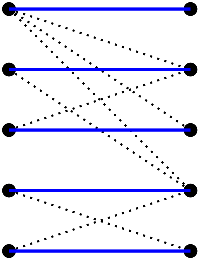


# Approximate Maximum Matching

- A matching is a set of vertex disjoint edges in a graph.

## Problem

Compute a matching which is as large as possible (compared to a maximum matching).

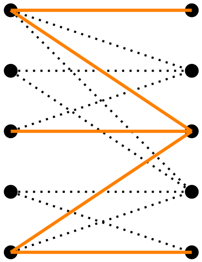


# Approximate Maximum Matching

- A matching is a set of vertex disjoint edges in a graph.

## Problem

Compute a matching which is as large as possible (compared to a maximum matching).



# The Model

# The Model

- **Q:** How do we partition a graph between two players?

# The Model

- **Q:** How do we partition a graph between two players?
- **A:** We partition the edges.

# The Model

- **Q:** How do we partition a graph between two players?
- **A:** We partition the edges.

### Adversarial Model

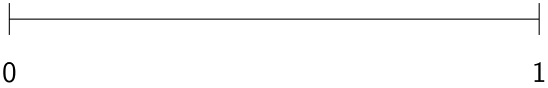
Edges are partitioned adversarially.

# The Model

- **Q:** How do we partition a graph between two players?
- **A:** We partition the edges.

## Adversarial Model

Edges are partitioned adversarially.

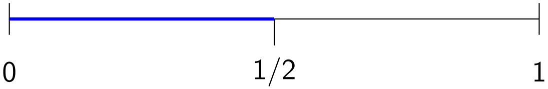


# The Model

- **Q:** How do we partition a graph between two players?
- **A:** We partition the edges.

## Adversarial Model

Edges are partitioned adversarially.

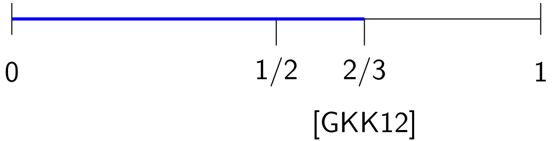


# The Model

- **Q:** How do we partition a graph between two players?
- **A:** We partition the edges.

## Adversarial Model

Edges are partitioned adversarially.

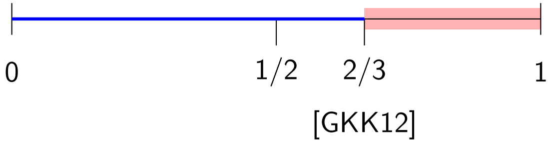


# The Model

- **Q:** How do we partition a graph between two players?
- **A:** We partition the edges.

## Adversarial Model

Edges are partitioned adversarially.



# The Model

- **Q:** How do we partition a graph between two players?
- **A:** We partition the edges.

# The Model

- **Q:** How do we partition a graph between two players?
- **A:** We partition the edges.

### Robust Model

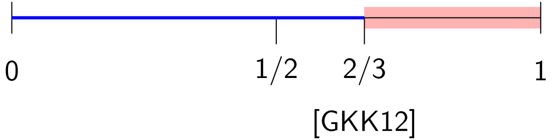
Edges are partitioned uniformly randomly.

# The Model

- **Q:** How do we partition a graph between two players?
- **A:** We partition the edges.

## Robust Model

Edges are partitioned uniformly randomly.

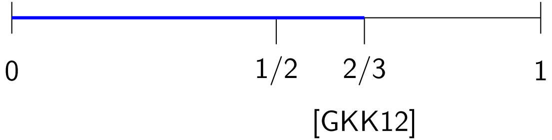


# The Model

- **Q:** How do we partition a graph between two players?
- **A:** We partition the edges.

## Robust Model

Edges are partitioned uniformly randomly.

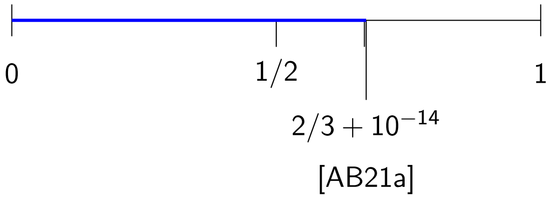


# The Model

- **Q:** How do we partition a graph between two players?
- **A:** We partition the edges.

## Robust Model

Edges are partitioned uniformly randomly.

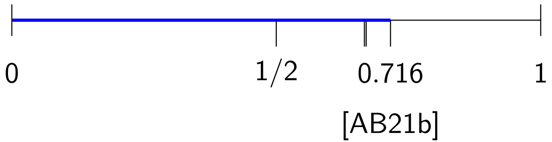


# The Model

- **Q:** How do we partition a graph between two players?
- **A:** We partition the edges.

## Robust Model

Edges are partitioned uniformly randomly.

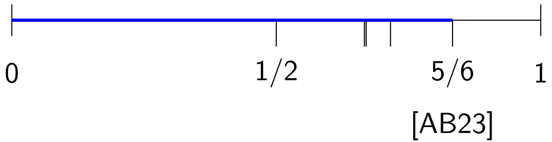


# The Model

- **Q:** How do we partition a graph between two players?
- **A:** We partition the edges.

## Robust Model

Edges are partitioned uniformly randomly.



# Warmup: Greedy as a Protocol

# Warmup: Greedy as a Protocol

- **Idea:** What if Alice runs a greedy algorithm?

## Warmup: Greedy as a Protocol

- **Idea:** What if Alice runs a greedy algorithm?

Theorem

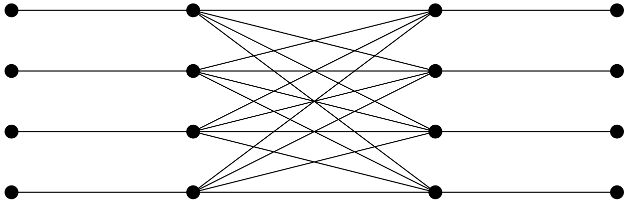
*A protocol which runs greedy computes a* $5/8$*-approximation in expectation, and this is tight.*

# Warmup: Greedy as a Protocol

- **Idea:** What if Alice runs a greedy algorithm?

### Theorem

*A protocol which runs greedy computes a* 5/8*-approximation in expectation, and this is tight.*

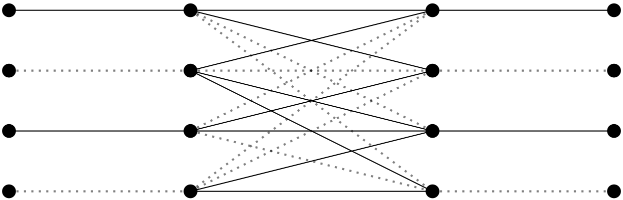


# Warmup: Greedy as a Protocol

- **Idea:** What if Alice runs a greedy algorithm?

### Theorem

*A protocol which runs greedy computes a* $5/8$*-approximation in expectation, and this is tight.*

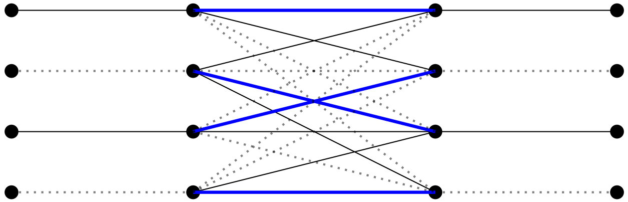


# Warmup: Greedy as a Protocol

- **Idea:** What if Alice runs a greedy algorithm?

### Theorem

*A protocol which runs greedy computes a* 5/8*-approximation in expectation, and this is tight.*

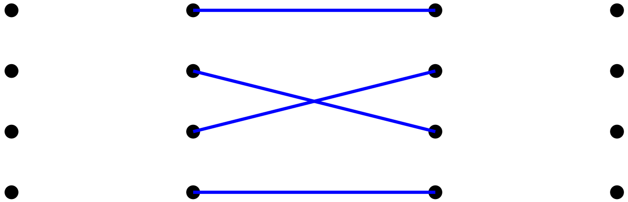


# Warmup: Greedy as a Protocol

- **Idea:** What if Alice runs a greedy algorithm?

**Theorem**

*A protocol which runs greedy computes a* 5/8*-approximation in expectation, and this is tight.*

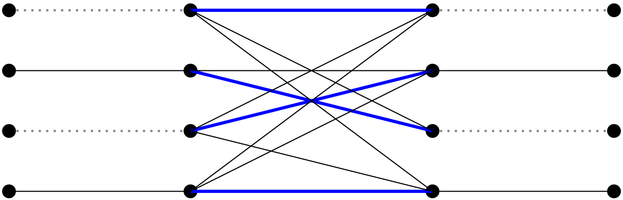


# Warmup: Greedy as a Protocol

- **Idea:** What if Alice runs a greedy algorithm?

### Theorem

*A protocol which runs greedy computes a* 5/8*-approximation in expectation, and this is tight.*

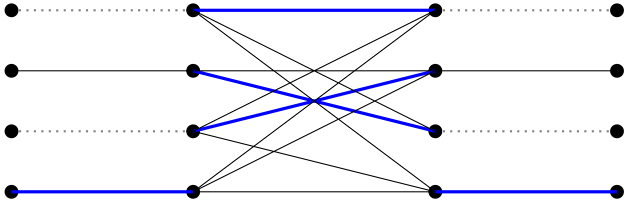


# Warmup: Greedy as a Protocol

- **Idea:** What if Alice runs a greedy algorithm?

### Theorem

*A protocol which runs greedy computes a 5/8-approximation in expectation, and this is tight.*

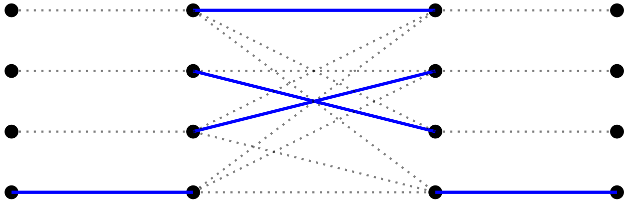


# Warmup: Greedy as a Protocol

- **Idea:** What if Alice runs a greedy algorithm?

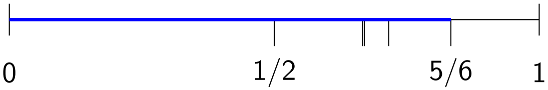
**Theorem**

*A protocol which runs greedy computes a* $5/8$*-approximation in expectation, and this is tight.*

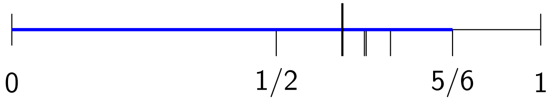


# Previous Work

# Previous Work



# Previous Work



# Maximum Matching as a Protocol

# Maximum Matching as a Protocol

- We need a protocol which is more clever.

# Maximum Matching as a Protocol

- We need a protocol which is more clever.
- **Idea:** What if Alice computes a maximum matching on their edges?

# Maximum Matching as a Protocol

- We need a protocol which is more clever.
- **Idea:** What if Alice computes a maximum matching on their edges?

### Theorem

*A protocol which sends a maximum matching computes a* $2/3$*-approximation in expectation.*

# Maximum Matching as a Protocol

- We need a protocol which is more clever.
- **Idea:** What if Alice computes a maximum matching on their edges?

**Theorem**

*A protocol which sends a maximum matching computes a* $2/3$*-approximation in expectation.*

**Theorem**

*A protocol which sends a maximum matching cannot exceed a* $5/6$*-approximation in expectation.*

# Maximum Matching as a Protocol

# Maximum Matching as a Protocol

**Theorem**

*A protocol which computes a maximum matching computes a* $2/3$*-approximation in expectation.*

# Maximum Matching as a Protocol

**Theorem**
*A protocol which computes a maximum matching computes a* 2/3*-approximation in expectation.*

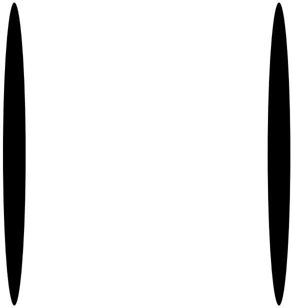
Proof Sketch:

# Maximum Matching as a Protocol

**Theorem**

*A protocol which computes a maximum matching computes a 2/3-approximation in expectation.*

Proof Sketch:

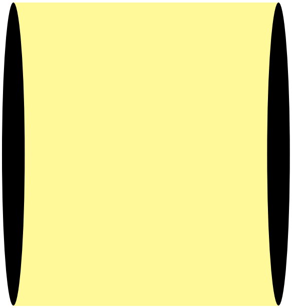


# Maximum Matching as a Protocol

**Theorem**

*A protocol which computes a maximum matching computes a 2/3-approximation in expectation.*

Proof Sketch:



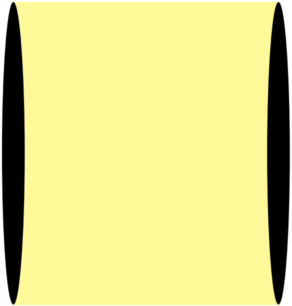
# Maximum Matching as a Protocol

### Theorem

*A protocol which computes a maximum matching computes a 2/3-approximation in expectation.*

Proof Sketch:

1. Pick a maximum matching $OPT$ in $G$.



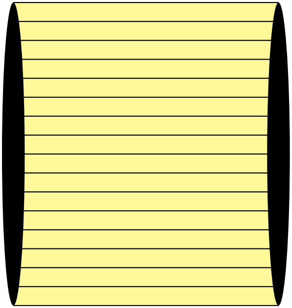
# Maximum Matching as a Protocol

**Theorem**

*A protocol which computes a maximum matching computes a $2/3$-approximation in expectation.*

Proof Sketch:

1. Pick a maximum matching $OPT$ in $G$.



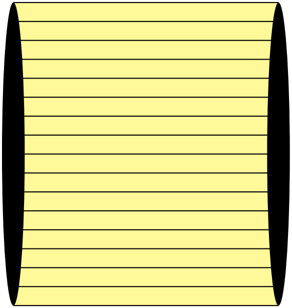
# Maximum Matching as a Protocol

**Theorem**

*A protocol which computes a maximum matching computes a 2/3-approximation in expectation.*

Proof Sketch:

1. Pick a maximum matching $OPT$ in $G$.
2. In expectation, Alice holds half of these.



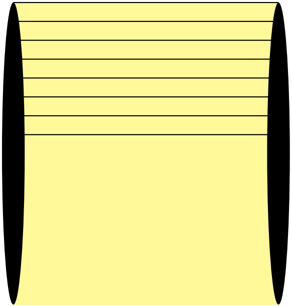
# Maximum Matching as a Protocol

**Theorem**

*A protocol which computes a maximum matching computes a 2/3-approximation in expectation.*

Proof Sketch:

1. Pick a maximum matching $OPT$ in $G$.
2. In expectation, Alice holds half of these.



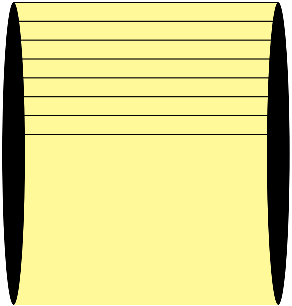
# Maximum Matching as a Protocol

## Theorem

*A protocol which computes a maximum matching computes a 2/3-approximation in expectation.*

Proof Sketch:

1. Pick a maximum matching $OPT$ in $G$.
2. In expectation, Alice holds half of these.
3. **Assumption:** Alice holds a $(\frac{1}{2} + \epsilon)$-approximation.



# Maximum Matching as a Protocol

**Theorem**

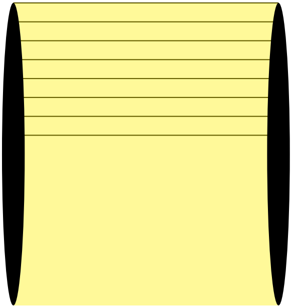
*A protocol which computes a maximum matching computes a $2/3$-approximation in expectation.*

# Maximum Matching as a Protocol

**Theorem**

*A protocol which computes a maximum matching computes a 2/3-approximation in expectation.*

3. **Assumption:** Alice holds a $(\frac{1}{2} + \epsilon)$-approximation.

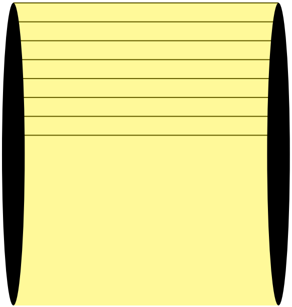


# Maximum Matching as a Protocol

**Theorem**

*A protocol which computes a maximum matching computes a* $2/3$*-approximation in expectation.*

3. **Assumption:** Alice holds a $(\frac{1}{2} + \epsilon)$-approximation.
4. Alice computes a maximum matching.

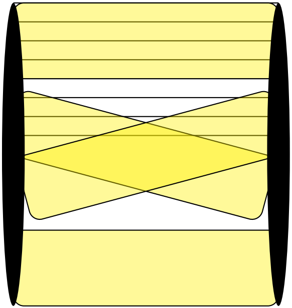


# Maximum Matching as a Protocol

### Theorem

*A protocol which computes a maximum matching computes a 2/3-approximation in expectation.*

3. **Assumption:** Alice holds a $(\frac{1}{2}+\epsilon)$-approximation.
4. Alice computes a maximum matching.

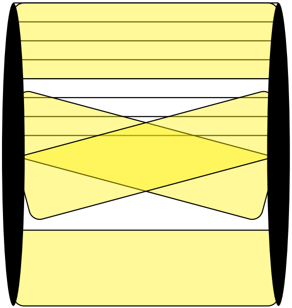


# Maximum Matching as a Protocol

### Theorem

*A protocol which computes a maximum matching computes a 2/3-approximation in expectation.*

3. **Assumption:** Alice holds a $(\frac{1}{2} + \epsilon)$-approximation.
4. Alice computes a maximum matching.
5. **Lemma:** There is a subgraph which contains at most a $\epsilon$-approximation.

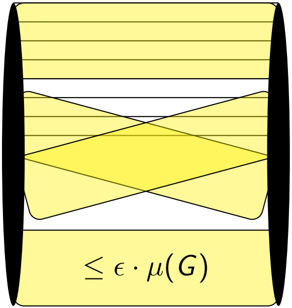


# Maximum Matching as a Protocol

**Theorem**

*A protocol which computes a maximum matching computes a $2/3$-approximation in expectation.*

3. **Assumption:** Alice holds a $(\frac{1}{2}+\epsilon)$-approximation.
4. Alice computes a maximum matching.
5. **Lemma:** There is a subgraph which contains at most a $\epsilon$-approximation.

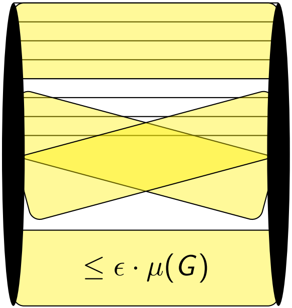


# Maximum Matching as a Protocol

### Theorem

*A protocol which computes a maximum matching computes a* 2/3*-approximation in expectation.*

3. **Assumption:** Alice holds a $(\frac{1}{2} + \epsilon)$-approximation.
4. Alice computes a maximum matching.
5. **Lemma:** There is a subgraph which contains at most a $\epsilon$-approximation.
6. By the random partitioning, the other regions cannot be larger.

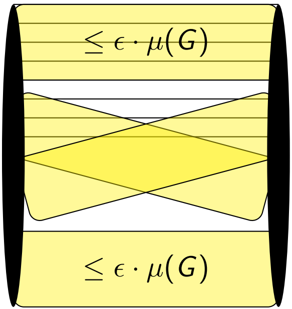


# Maximum Matching as a Protocol

## Theorem

*A protocol which computes a maximum matching computes a 2/3-approximation in expectation.*

3. **Assumption:** Alice holds a $(\frac{1}{2}+\epsilon)$-approximation.
4. Alice computes a maximum matching.
5. **Lemma:** There is a subgraph which contains at most a $\epsilon$-approximation.
6. By the random partitioning, the other regions cannot be larger.



# Maximum Matching as a Protocol

**Theorem**

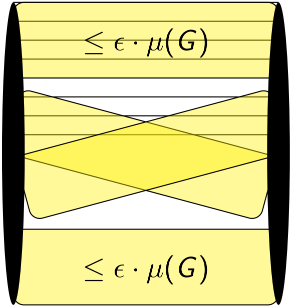
*A protocol which computes a maximum matching computes a* $2/3$*-approximation in expectation.*

# Maximum Matching as a Protocol

## Theorem

*A protocol which computes a maximum matching computes a* $2/3$*-approximation in expectation.*

6. By the random partitititoning, the other regions cannot be larger.

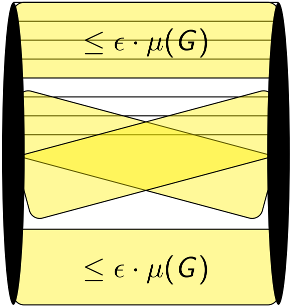


# Maximum Matching as a Protocol

### Theorem

*A protocol which computes a maximum matching computes a* 2/3*-approximation in expectation.*

6. By the random partititoning, the other regions cannot be larger.
7. Send the message to Bob.

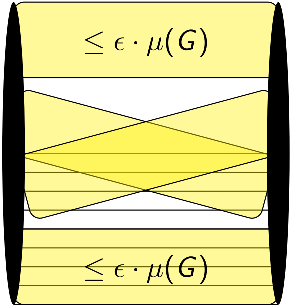


# Maximum Matching as a Protocol

**Theorem**

*A protocol which computes a maximum matching computes a 2/3-approximation in expectation.*

6. By the random partititoning, the other regions cannot be larger.
7. Send the message to Bob.

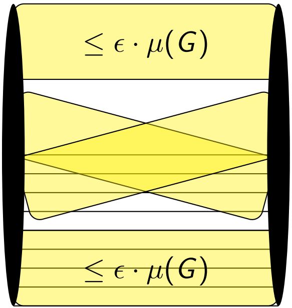


# Maximum Matching as a Protocol

**Theorem**

*A protocol which computes a maximum matching computes a $2/3$-approximation in expectation.*

6. By the random partititoning, the other regions cannot be larger.
7. Send the message to Bob.
8. Bob then holds a $(1-2\epsilon)$-approximation.

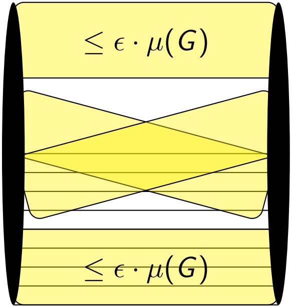


# Maximum Matching as a Protocol

### Theorem

*A protocol which computes a maximum matching computes a* 2/3*-approximation in expectation.*

6. By the random partitittoning, the other regions cannot be larger.
7. Send the message to Bob.
8. Bob then holds a $(1-2\epsilon)$-approximation.
9. Solve $(1-2\epsilon) = (\frac{1}{2}+\epsilon)$.



# The Big Question

# The Big Question

**Question**

Can we close the gap between $2/3$ and $5/6$?

# The Big Question

### Question

Can we close the gap between $2/3$ and $5/6$?

### Question

Do these techniques help us go beyond $5/6$?

# References I


Sepehr Assadi and Soheil Behnezhad.
Beating Two-Thirds For Random-Order Streaming Matching.
In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, 48th International Colloquium on Automata, Languages, and Programming (ICALP 2021), volume 198 of Leibniz International Proceedings in Informatics (LIPIcs), pages 19:1–19:13, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Sepehr Assadi and Soheil Behnezhad.
On the Robust Communication Complexity of Bipartite Matching.
In Mary Wootters and Laura Sanità, editors, Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021), volume 207 of Leibniz International Proceedings in Informatics (LIPIcs),

# References II

pages 48:1–48:17, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Amir Azarmehr and Soheil Behnezhad.
Robust Communication Complexity of Matching: EDCS Achieves 5/6 Approximation.
In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, 50th International Colloquium on Automata, Languages, and Programming (ICALP 2023), volume 261 of Leibniz International Proceedings in Informatics (LIPIcs), pages 14:1–14:15, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

# References III

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