Constructing Large Matchings via Query Access to a Maximal Matching Oracle

FSTTCS 2020

Lidiya Khalidah binti Khalil and Christian Konrad

University of BRISTOL

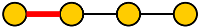
Matchings

Let $G = (V, E)$ be a graph

Matchings

Let $G = (V, E)$ be a graph

- A *matching* $M \subseteq E$ is a subset of non-adjacent edges



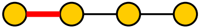
matching

not a matching

Matchings

Let $G = (V, E)$ be a graph

- A *matching* $M \subseteq E$ is a subset of non-adjacent edges

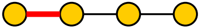


matching

not a matching

- M is *maximal* if $M \cup \{e\}$ is not a matching, for every $e \in E \setminus M$

maximal

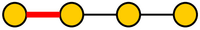


not maximal

Matchings

Let $G = (V, E)$ be a graph

- A *matching* $M \subseteq E$ is a subset of non-adjacent edges

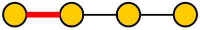


matching

not a matching

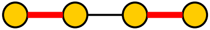
- M is *maximal* if $M \cup \{e\}$ is not a matching, for every $e \in E \setminus M$

maximal



not maximal

- M^* is *maximum* if for every other matching $M \subseteq E$: $|M^*| \geq |M|$

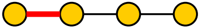


maximum

Matchings

Let $G = (V, E)$ be a graph

- A *matching* $M \subseteq E$ is a subset of non-adjacent edges

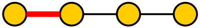


matching

not a matching

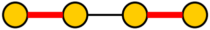
- M is *maximal* if $M \cup \{e\}$ is not a matching, for every $e \in E \setminus M$

maximal



not maximal

- M^* is *maximum* if for every other matching $M \subseteq E$: $|M^*| \geq |M|$



maximum

Property: $|\text{maximal matching}| \geq \frac{1}{2} \, |\text{maximum matching}|$

Computing Maximal Matchings is often easy

Goal: Maximum Matching approximation (better than 1/2-approx.)

Computing Maximal Matchings is often easy

Goal: Maximum Matching approximation (better than 1/2-approx.)

In many computational models... (e.g. streaming, distributed models)

- computing maximal matchings is easy
- computing maximum matching approximations is more difficult

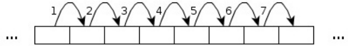
Computing Maximal Matchings is often easy

Goal: Maximum Matching approximation (better than 1/2-approx.)

In many computational models... (e.g. streaming, distributed models)

- computing maximal matchings is easy
- computing maximum matching approximations is more difficult

Edge-arrival Streaming Model:



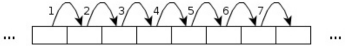
Computing Maximal Matchings is often easy

Goal: Maximum Matching approximation (better than 1/2-approx.)

In many computational models... (e.g. streaming, distributed models)

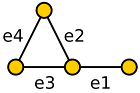
- computing maximal matchings is easy
- computing maximum matching approximations is more difficult

Edge-arrival Streaming Model:



- Input stream: Sequence of edges of input graph $G = (V, E)$ with $n = |V|$ in arbitrary order

$$S = e_2 e_1 e_4 e_3$$



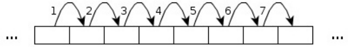
Computing Maximal Matchings is often easy

Goal: Maximum Matching approximation (better than 1/2-approx.)

In many computational models... (e.g. streaming, distributed models)

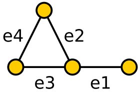
- computing maximal matchings is easy
- computing maximum matching approximations is more difficult

Edge-arrival Streaming Model:



- Input stream: Sequence of edges of input graph $G = (V, E)$ with $n = |V|$ in arbitrary order

$$S = e_2 e_1 e_4 e_3$$



- Goal: Few passes algorithms with small space

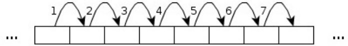
Computing Maximal Matchings is often easy

Goal: Maximum Matching approximation (better than 1/2-approx.)

In many computational models... (e.g. streaming, distributed models)

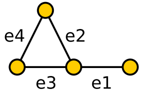
- computing maximal matchings is easy
- computing maximum matching approximations is more difficult

Edge-arrival Streaming Model:



- Input stream: Sequence of edges of input graph $G = (V, E)$ with $n = |V|$ in arbitrary order

$$S = e_2 e_1 e_4 e_3$$



- Goal: Few passes algorithms with small space
- Streaming Maximal Matching Algorithm: Insert current edge into initially empty matching if possible ($\textsc{Greedy}$), using space $\tilde{O}(n)$

State of the Art Streaming Matching Algorithms

# passes	Approximation	det/rand	Reference
Bipartite Graphs			
1	$\frac{1}{2}$	det	Greedy, folklore
2	$2-\sqrt{2} \approx 0.5857$	rand	Konrad '18
3	0.6067	rand	Konrad '18
$O(\frac{1}{\epsilon^2})$	$1-\epsilon$	det	Assadi, Liu, Tarjan '21
General Graphs			
1	$\frac{1}{2}$	det	Greedy, folklore
2	0.53125	det	Kale and Tirodkar '17
$\frac{1}{\epsilon}^{O(\frac{1}{\epsilon})}$	$1-\epsilon$	det	Tirodkar '18

State of the Art Streaming Matching Algorithms

# passes	Approximation	det/rand	Reference
Bipartite Graphs			
1	$\frac{1}{2}$	det	Greedy, folklore
2	$2-\sqrt{2} \approx 0.5857$	rand	Konrad '18
3	0.6067	rand	Konrad '18
$O(\frac{1}{\epsilon^2})$	$1-\epsilon$	det	Assadi, Liu, Tarjan '21
General Graphs			
1	$\frac{1}{2}$	det	Greedy, folklore
2	0.53125	det	Kale and Tirodkar '17
$\frac{1}{\epsilon}^{O(\frac{1}{\epsilon})}$	$1-\epsilon$	det	Tirodkar '18

Most of these algorithms (including previous works) solely run Greedy in carefully selected subgraphs in each pass, thereby collecting edges and outputting the largest matching among the edges stored.

State of the Art Streaming Matching Algorithms

# passes	Approximation	det/rand	Reference
Bipartite Graphs			
1	$\frac{1}{2}$	det	GREEDY, folklore
2	$2-\sqrt{2} \approx 0.5857$	rand	Konrad '18
3	0.6067	rand	Konrad '18
$\mathrm{O}(\frac{1}{\epsilon^2})$	$1-\epsilon$	det	Assadi, Liu, Tarjan '21
General Graphs			
1	$\frac{1}{2}$	det	GREEDY, folklore
2	0.53125	det	Kale and Tirodkar '17
$\frac{1}{\epsilon}^{\mathrm{O}(\frac{1}{\epsilon})}$	$1-\epsilon$	det	Tirodkar '18

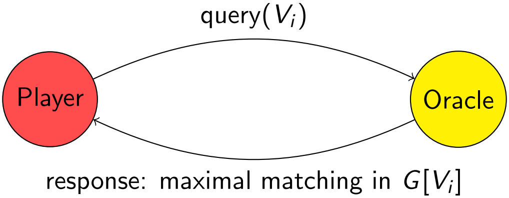
Most of these algorithms (including previous works) solely run GREEDY in carefully selected subgraphs in each pass, thereby collecting edges and outputting the largest matching among the edges stored.

How large a matching can we compute if we solely invoke Greedy in each pass?

Maximal Matching Oracle

Maximal Matching Oracle

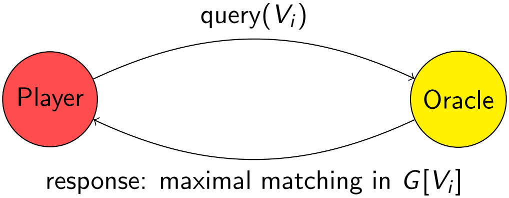
Matching Game:



- Player and oracle play r rounds of a "matching game"
- In each round r:
  1. Player queries a subset of vertices $V_i \subseteq V$
  2. Oracle returns maximal matching M_i in induced subgraph $G[V_i]$
- Player outputs largest matching in $\cup_{1 \leq i \leq r} M_i$

Maximal Matching Oracle

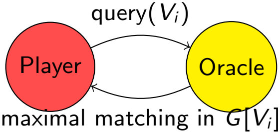
Matching Game:



- Player and oracle play r rounds of a "matching game"
- In each round r:
  1. Player queries a subset of vertices $V_i \subseteq V$
  2. Oracle returns maximal matching M_i in induced subgraph $G[V_i]$
- Player outputs largest matching in $\cup_{1 \leq i \leq r} M_i$

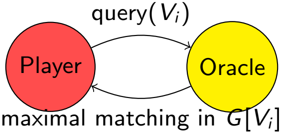
Research Question: What is the trade-off between the number of rounds and the approximation ratio?

Matching Game - Upper Bounds



Upper Bounds for Bipartite Graphs:

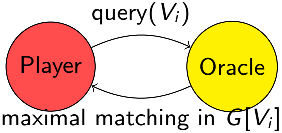
Matching Game - Upper Bounds



Upper Bounds for Bipartite Graphs:

- **1 round:** $\frac{1}{2}$-approximation
  query(V) yields maximal matching in input graph

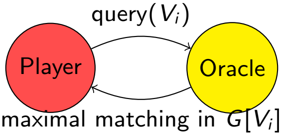
Matching Game - Upper Bounds



Upper Bounds for Bipartite Graphs:

- **1 round:** $\frac{1}{2}$-approximation
  query(V) yields maximal matching in input graph
- **2 rounds:** $\geq \frac{1}{2}$-approximation

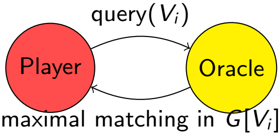
Matching Game - Upper Bounds



Upper Bounds for Bipartite Graphs:

- **1 round:** $\frac{1}{2}$-approximation
  query(V) yields maximal matching in input graph
- **2 rounds:** $\geq \frac{1}{2}$-approximation
- **3 rounds:** $3/5$-approximation
  3-pass streaming algorithm analysed by Kale and Tirodkar '17

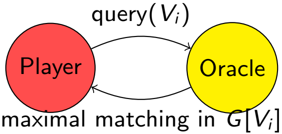
Matching Game - Upper Bounds



Upper Bounds for Bipartite Graphs:

- **1 round:** $\frac{1}{2}$-approximation
  query(V) yields maximal matching in input graph
- **2 rounds:** $\geq \frac{1}{2}$-approximation
- **3 rounds:** $3/5$-approximation
  3-pass streaming algorithm analysed by Kale and Tirodkar '17
- $\Theta(\frac{1}{\epsilon^6})$ **rounds:** $(1-\epsilon)$-approximation
  Streaming algorithm that runs in $\Theta(\frac{1}{\epsilon^5})$ passes by Eggert et al. '12 can be adapted to the model

Matching Game - Upper Bounds

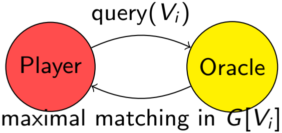


Upper Bounds for Bipartite Graphs:

- **1 round:** $\frac{1}{2}$-approximation
  query(V) yields maximal matching in input graph
- **2 rounds:** $\geq \frac{1}{2}$-approximation
- **3 rounds:** $3/5$-approximation
  3-pass streaming algorithm analysed by Kale and Tirodkar '17
- $\Theta(\frac{1}{\epsilon^6})$ **rounds:** $(1-\epsilon)$-approximation
  Streaming algorithm that runs in $\Theta(\frac{1}{\epsilon^5})$ passes by Eggert et al. '12 can be adapted to the model

Upper Bounds for General Graphs:

Matching Game - Upper Bounds



Upper Bounds for Bipartite Graphs:

- **1 round:** $\frac{1}{2}$-approximation
  query(V) yields maximal matching in input graph
- **2 rounds:** $\geq \frac{1}{2}$-approximation
- **3 rounds:** $3/5$-approximation
  3-pass streaming algorithm analysed by Kale and Tirodkar '17
- $\Theta(\frac{1}{\epsilon^6})$ **rounds:** $(1 - \epsilon)$-approximation
  Streaming algorithm that runs in $\Theta(\frac{1}{\epsilon^5})$ passes by Eggert et al. '12 can be adapted to the model

Upper Bounds for General Graphs: ✗ (except 1 round $\frac{1}{2}$-approx.)

Our Results

We give the following Lower Bound Results:

Our Results

We give the following Lower Bound Results:

Bipartite Graphs

- 1 round $\frac{1}{2}$-approximation:

Our Results

We give the following Lower Bound Results:

Bipartite Graphs

- 1 round $\frac{1}{2}$-approximation: optimal ✓

Our Results

We give the following Lower Bound Results:

Bipartite Graphs

- 1 round $\frac{1}{2}$-approximation: optimal ✓
- 2 rounds $\geq \frac{1}{2}$-approximation:

Our Results

We give the following Lower Bound Results:

Bipartite Graphs

- 1 round $\frac{1}{2}$-approximation: optimal ✓
- 2 rounds $\geq \frac{1}{2}$-approximation: $\frac{1}{2}$ is best possible ✓

Our Results

We give the following Lower Bound Results:

Bipartite Graphs

- 1 round $\frac{1}{2}$-approximation: optimal ✓
- 2 rounds $\geq \frac{1}{2}$-approximation: $\frac{1}{2}$ is best possible ✓
- 3 rounds $\frac{3}{5}$-approximation:

Our Results

We give the following Lower Bound Results:

Bipartite Graphs

- 1 round $\frac{1}{2}$-approximation: optimal ✓
- 2 rounds $\geq \frac{1}{2}$-approximation: $\frac{1}{2}$ is best possible ✓
- 3 rounds $\frac{3}{5}$-approximation: optimal ✓

Our Results

We give the following Lower Bound Results:

Bipartite Graphs

- 1 round $\frac{1}{2}$-approximation: optimal ✓
- 2 rounds $\geq \frac{1}{2}$-approximation: $\frac{1}{2}$ is best possible ✓
- 3 rounds $\frac{3}{5}$-approximation: optimal ✓
- $\Theta(\frac{1}{\epsilon^6})$ rounds $(1-\epsilon)$-approximation:

Our Results

We give the following Lower Bound Results:

Bipartite Graphs

- 1 round $\frac{1}{2}$-approximation: optimal ✓
- 2 rounds $\geq \frac{1}{2}$-approximation: $\frac{1}{2}$ is best possible ✓
- 3 rounds $\frac{3}{5}$-approximation: optimal ✓
- $\Theta(\frac{1}{\epsilon^6})$ rounds $(1-\epsilon)$-approximation: $\Omega(\frac{1}{\epsilon})$ rounds are needed for a $(1-\epsilon)$-approximation

Our Results

We give the following Lower Bound Results:

Bipartite Graphs

- 1 round $\frac{1}{2}$-approximation: optimal ✓
- 2 rounds $\geq \frac{1}{2}$-approximation: $\frac{1}{2}$ is best possible ✓
- 3 rounds $\frac{3}{5}$-approximation: optimal ✓
- $\Theta(\frac{1}{\epsilon^6})$ rounds $(1-\epsilon)$-approximation: $\Omega(\frac{1}{\epsilon})$ rounds are needed for a $(1-\epsilon)$-approximation

General Graphs (1 round 1/2-approximation)

Our Results

We give the following Lower Bound Results:

Bipartite Graphs

- 1 round $\frac{1}{2}$-approximation: optimal ✓
- 2 rounds $\geq \frac{1}{2}$-approximation: $\frac{1}{2}$ is best possible ✓
- 3 rounds $\frac{3}{5}$-approximation: optimal ✓
- $\Theta(\frac{1}{\epsilon^6})$ rounds $(1-\epsilon)$-approximation: $\Omega(\frac{1}{\epsilon})$ rounds are needed for a $(1-\epsilon)$-approximation

General Graphs (1 round 1/2-approximation)
$\Omega(n)$ rounds are needed for an approximation ratio $\frac{1}{2}+\epsilon$, for any $\epsilon > 0$

Our Results

We give the following Lower Bound Results:

Bipartite Graphs

- 1 round $\frac{1}{2}$-approximation: optimal ✓
- 2 rounds $\geq \frac{1}{2}$-approximation: $\frac{1}{2}$ is best possible ✓
- 3 rounds $\frac{3}{5}$-approximation: optimal ✓
- $\Theta(\frac{1}{\epsilon^6})$ rounds $(1-\epsilon)$-approximation: $\Omega(\frac{1}{\epsilon})$ rounds are needed for a $(1-\epsilon)$-approximation

General Graphs (1 round 1/2-approximation)
$\Omega(n)$ rounds are needed for an approximation ratio $\frac{1}{2}+\epsilon$, for any $\epsilon > 0$

Outline:

1. $\Omega(\frac{1}{\epsilon})$ rounds are needed for a $(1-\epsilon)$-approximation
2. $\Omega(n)$ rounds needed for $(\frac{1}{2}+\epsilon)$-approx. in general graphs, $\epsilon > 0$
3. 0.6-approximation lower bound for 3 rounds (main technical result)

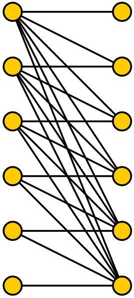
Lower Bound for Computing a $(1 - \epsilon)$-approximation

Theorem. Any query algorithm with approximation factor $1 - \epsilon$ requires at least $\frac{1}{\epsilon} - 1$ queries,even in bipartite graphs.

Lower Bound for Computing a $(1-\epsilon)$-approximation

Theorem. Any query algorithm with approximation factor $1-\epsilon$ requires at least $\frac{1}{\epsilon}-1$ queries,even in bipartite graphs.

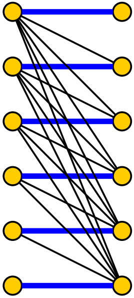
Proof. Consider semi-complete graph on $2c$ vertices



Lower Bound for Computing a $(1-\epsilon)$-approximation

Theorem. Any query algorithm with approximation factor $1-\epsilon$ requires at least $\frac{1}{\epsilon}-1$ queries,even in bipartite graphs.

Proof. Consider semi-complete graph on $2c$ vertices

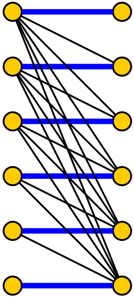


- Unique perfect matching M^* of size c

Lower Bound for Computing a $(1-\epsilon)$-approximation

Theorem. Any query algorithm with approximation factor $1-\epsilon$ requires at least $\frac{1}{\epsilon}-1$ queries,even in bipartite graphs.

Proof. Consider semi-complete graph on $2c$ vertices

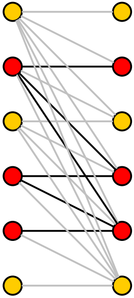


- Unique perfect matching M^* of size c
- Sub-optimal matching constitutes at best a $\frac{c-1}{c}=(1-\frac{1}{c})$-approximation

Lower Bound for Computing a $(1-\epsilon)$-approximation

Theorem. Any query algorithm with approximation factor $1-\epsilon$ requires at least $\frac{1}{\epsilon} - 1$ queries,even in bipartite graphs.

Proof. Consider semi-complete graph on $2c$ vertices

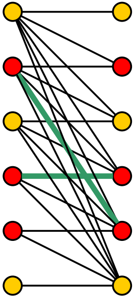


- Unique perfect matching M^* of size c
- Sub-optimal matching constitutes at best a $\frac{c-1}{c} = (1 - \frac{1}{c})$-approximation
- **Insight:** Any query gives at most one edge from M^*

Lower Bound for Computing a $(1-\epsilon)$-approximation

Theorem. Any query algorithm with approximation factor $1-\epsilon$ requires at least $\frac{1}{\epsilon}-1$ queries,even in bipartite graphs.

Proof. Consider semi-complete graph on $2c$ vertices

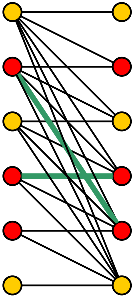


- Unique perfect matching M^* of size c
- Sub-optimal matching constitutes at best a $\frac{c-1}{c} = (1-\frac{1}{c})$-approximation
- **Insight:** Any query gives at most one edge from M^*

Lower Bound for Computing a $(1 - \epsilon)$-approximation

Theorem. Any query algorithm with approximation factor $1 - \epsilon$ requires at least $\frac{1}{\epsilon} - 1$ queries,even in bipartite graphs.

Proof. Consider semi-complete graph on $2c$ vertices

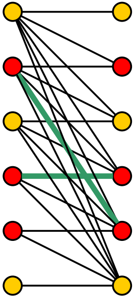


- Unique perfect matching M^* of size c
- Sub-optimal matching constitutes at best a $\frac{c-1}{c} = (1 - \frac{1}{c})$-approximation
- **Insight:** Any query gives at most one edge from M^*
- Hence, to achieve a $(1 - \epsilon)$-approximation, for $\epsilon = \frac{1}{c+1}$, $c = \frac{1}{\epsilon} - 1$ queries are needed

Lower Bound for Computing a $(1-\epsilon)$-approximation

Theorem. Any query algorithm with approximation factor $1-\epsilon$ requires at least $\frac{1}{\epsilon}-1$ queries,even in bipartite graphs.

Proof. Consider semi-complete graph on $2c$ vertices

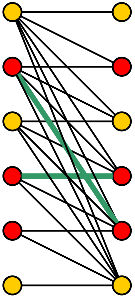


- Unique perfect matching M^* of size c
- Sub-optimal matching constitutes at best a $\frac{c-1}{c} = (1-\frac{1}{c})$-approximation
- **Insight:** Any query gives at most one edge from M^*
- Hence, to achieve a $(1-\epsilon)$-approximation, for $\epsilon = \frac{1}{c+1}$, $c = \frac{1}{\epsilon}-1$ queries are needed
- Use multiple disjoint gadgets for arbitrary n

Lower Bound for Computing a $(1-\epsilon)$-approximation

Theorem. Any query algorithm with approximation factor $1-\epsilon$ requires at least $\frac{1}{\epsilon}-1$ queries,even in bipartite graphs.

Proof. Consider semi-complete graph on $2c$ vertices



- Unique perfect matching M^* of size c
- Sub-optimal matching constitutes at best a $\frac{c-1}{c} = (1-\frac{1}{c})$-approximation
- **Insight:** Any query gives at most one edge from M^*
- Hence, to achieve a $(1-\epsilon)$-approximation, for $\epsilon = \frac{1}{c+1}$, $c = \frac{1}{\epsilon} - 1$ queries are needed
- Use multiple disjoint gadgets for arbitrary n

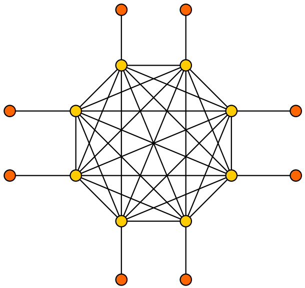
□

Lower Bound for General Graphs

Theorem. Any r-round query algorithm for general graphs has an approximation of at most $\frac{1}{2} + \frac{r}{n}$.

Theorem. Any r-round query algorithm for general graphs has an approximation of at most $\frac{1}{2} + \frac{r}{n}$.

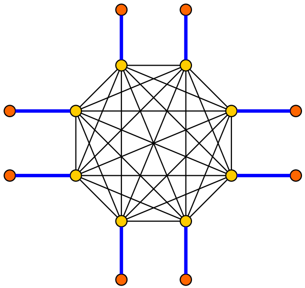
Proof. Consider a bomb graph on n vertices



Lower Bound for General Graphs

Theorem. Any r-round query algorithm for general graphs has an approximation of at most $\frac{1}{2} + \frac{r}{n}$.

Proof. Consider a bomb graph on n vertices

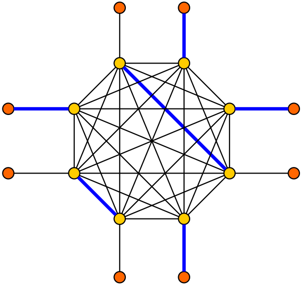


- "outside" edges = perfect matching

Lower Bound for General Graphs

Theorem. Any r-round query algorithm for general graphs has an approximation of at most $\frac{1}{2} + \frac{r}{n}$.

Proof. Consider a bomb graph on n vertices

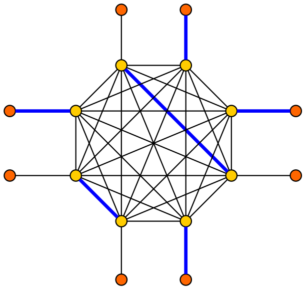


- "outside" edges = perfect matching
- "inside" edge: blocks two optimal edges

Lower Bound for General Graphs

Theorem. Any r-round query algorithm for general graphs has an approximation of at most $\frac{1}{2} + \frac{r}{n}$.

Proof. Consider a bomb graph on n vertices

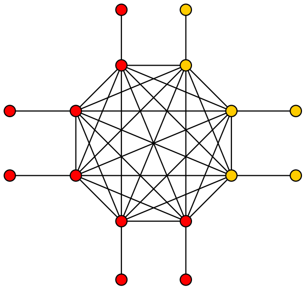


- "outside" edges = perfect matching
- "inside" edge: blocks two optimal edges
- $(\frac{1}{2} + \frac{r}{n})$-approx: r "outside" edges

Lower Bound for General Graphs

Theorem. Any r-round query algorithm for general graphs has an approximation of at most $\frac{1}{2} + \frac{r}{n}$.

Proof. Consider a bomb graph on n vertices

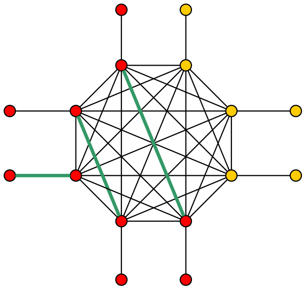


- "outside" edges = perfect matching
- "inside" edge: blocks two optimal edges
- $(\frac{1}{2} + \frac{r}{n})$-approx: r "outside" edges
- **Insight:** ≤ 1 "outside" edges per query

Lower Bound for General Graphs

Theorem. Any r-round query algorithm for general graphs has an approximation of at most $\frac{1}{2} + \frac{r}{n}$.

Proof. Consider a bomb graph on n vertices

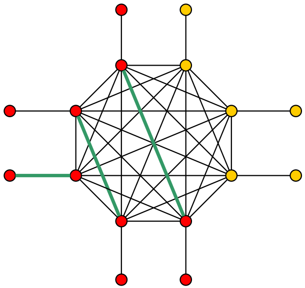


- "outside" edges = perfect matching
- "inside" edge: blocks two optimal edges
- $(\frac{1}{2} + \frac{r}{n})$-approx: r "outside" edges
- **Insight:** ≤ 1 "outside" edges per query

Lower Bound for General Graphs

Theorem. Any r-round query algorithm for general graphs has an approximation of at most $\frac{1}{2} + \frac{r}{n}$.

Proof. Consider a bomb graph on n vertices

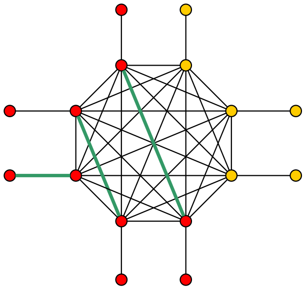


- "outside" edges = perfect matching
- "inside" edge: blocks two optimal edges
- $(\frac{1}{2} + \frac{r}{n})$-approx: r "outside" edges
- **Insight:** ≤ 1 "outside" edges per query
- r queries needed

Lower Bound for General Graphs

Theorem. Any r-round query algorithm for general graphs has an approximation of at most $\frac{1}{2} + \frac{r}{n}$.

Proof. Consider a bomb graph on n vertices



- "outside" edges = perfect matching
- "inside" edge: blocks two optimal edges
- $(\frac{1}{2} + \frac{r}{n})$-approx: r "outside" edges
- **Insight:** ≤ 1 "outside" edges per query
- r queries needed

□

Remarks

Deterministic / Randomized Query Algorithms:

- Lower bounds on previous slides hold even if the input graph is known by the player
- They also hold for randomized query algorithms

Lower Bound for 3 Rounds on Bipartite Graphs:

- More subtle argument
- Oracle builds graph that depends on the queries
- Lower bound therefore only holds for deterministic algorithms

Three Round Query Algorithm for Bipartite Graphs

Algorithm (input graph $G = (A, B, E)$)

Three Round Query Algorithm for Bipartite Graphs

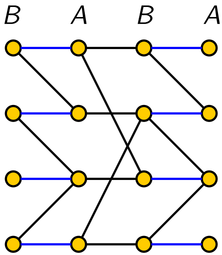
Algorithm (input graph $G = (A, B, E)$)

1. $M \leftarrow \text{query}(A \cup B)$
2. $M_L \leftarrow \text{query}(M(A) \cup \overline{M(B)})$
3. $B' \subseteq B(M) \leftarrow$ endpoints of path of length two in $M \cup M_L$
4. $M_R \leftarrow \text{query}(B' \cup \overline{M(A)})$
5. **return** largest matching using edges $M \cup M_L \cup M_R$

Three Round Query Algorithm for Bipartite Graphs

Algorithm (input graph $G = (A, B, E)$)

1. $M \leftarrow \text{query}(A \cup B)$
2. $M_L \leftarrow \text{query}(M(A) \cup \overline{M(B)})$
3. $B' \subseteq B(M) \leftarrow$ endpoints of path of length two in $M \cup M_L$
4. $M_R \leftarrow \text{query}(B' \cup \overline{M(A)})$
5. **return** largest matching using edges $M \cup M_L \cup M_R$

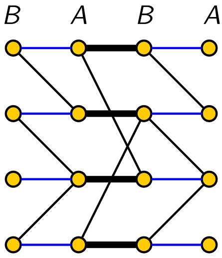


Input graph $G = (A, B, E)$ with perfect matching M^*

Three Round Query Algorithm for Bipartite Graphs

Algorithm (input graph $G = (A, B, E)$)

1. $M \leftarrow \text{query}(A \cup B)$
2. $M_L \leftarrow \text{query}(M(A) \cup \overline{M(B)})$
3. $B' \subseteq B(M) \leftarrow$ endpoints of path of length two in $M \cup M_L$
4. $M_R \leftarrow \text{query}(B' \cup \overline{M(A)})$
5. **return** largest matching using edges $M \cup M_L \cup M_R$

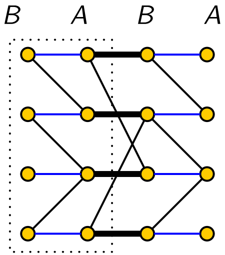


1st query: Matching M

Three Round Query Algorithm for Bipartite Graphs

Algorithm (input graph $G = (A, B, E)$)

1. $M \leftarrow \text{query}(A \cup B)$
2. $M_L \leftarrow \text{query}(M(A) \cup \overline{M(B)})$
3. $B' \subseteq B(M) \leftarrow$ endpoints of path of length two in $M \cup M_L$
4. $M_R \leftarrow \text{query}(B' \cup \overline{M(A)})$
5. **return** largest matching using edges $M \cup M_L \cup M_R$

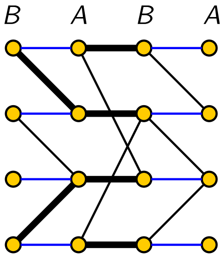


Subgraph $G[A(M) \cup \overline{M(B)}]$

Three Round Query Algorithm for Bipartite Graphs

Algorithm (input graph $G = (A, B, E)$)

1. $M \leftarrow \text{query}(A \cup B)$
2. $M_L \leftarrow \text{query}(M(A) \cup \overline{M(B)})$
3. $B' \subseteq B(M) \leftarrow$ endpoints of path of length two in $M \cup M_L$
4. $M_R \leftarrow \text{query}(B' \cup \overline{M(A)})$
5. **return** largest matching using edges $M \cup M_L \cup M_R$

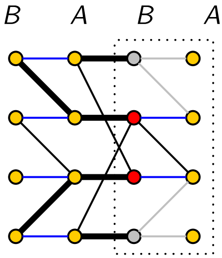


2nd query: Matching M_L

Three Round Query Algorithm for Bipartite Graphs

Algorithm (input graph $G = (A, B, E)$)

1. $M \leftarrow \text{query}(A \cup B)$
2. $M_L \leftarrow \text{query}(M(A) \cup \overline{M(B)})$
3. $B' \subseteq B(M) \leftarrow$ endpoints of path of length two in $M \cup M_L$
4. $M_R \leftarrow \text{query}(B' \cup \overline{M(A)})$
5. **return** largest matching using edges $M \cup M_L \cup M_R$

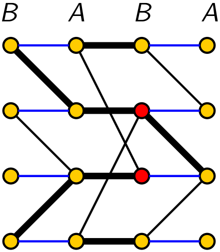


B' and Subgraph $G[\overline{A(M)} \cup B']$

Three Round Query Algorithm for Bipartite Graphs

Algorithm (input graph $G = (A, B, E)$)

1. $M \leftarrow \text{query}(A \cup B)$
2. $M_L \leftarrow \text{query}(M(A) \cup \overline{M(B)})$
3. $B' \subseteq B(M) \leftarrow$ endpoints of path of length two in $M \cup M_L$
4. $M_R \leftarrow \text{query}(B' \cup \overline{M(A)})$
5. **return** largest matching using edges $M \cup M_L \cup M_R$

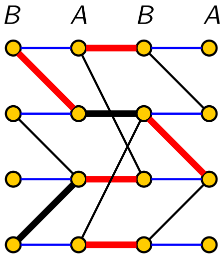


Matching M_R

Three Round Query Algorithm for Bipartite Graphs

Algorithm (input graph $G = (A, B, E)$)

1. $M \leftarrow \text{query}(A \cup B)$
2. $M_L \leftarrow \text{query}(M(A) \cup \overline{M(B)})$
3. $B' \subseteq B(M) \leftarrow$ endpoints of path of length two in $M \cup M_L$
4. $M_R \leftarrow \text{query}(B' \cup \overline{M(A)})$
5. **return** largest matching using edges $M \cup M_L \cup M_R$



Largest matching in $M \cup M_L \cup M_R$ (M augmented with $M_L \cup M_R$)

Three Round Query Algorithm for Bipartite Graphs (2)

Analysis:

Three Round Query Algorithm for Bipartite Graphs (2)

Analysis: $\frac{3}{5}$-approximation algorithm [Kale and Tirodkar, '17]

Three Round Query Algorithm for Bipartite Graphs (2)

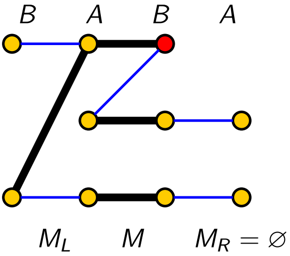
Analysis: $\frac{3}{5}$-approximation algorithm [Kale and Tirodkar, '17]

Worst-case Example:

Three Round Query Algorithm for Bipartite Graphs (2)

Analysis: $\frac{3}{5}$-approximation algorithm [Kale and Tirodkar, '17]

Worst-case Example:



1. $M \leftarrow \text{query}(A \cup B)$
2. $M_L \leftarrow \text{query}(M(A) \cup \overline{M(B)})$
3. $B' \subseteq B(M) \leftarrow$ endpoints of path of length two in $M \cup M_L$
4. $M_R \leftarrow \text{query}(B' \cup \overline{M(A)})$

Lower Bound Construction - First Query

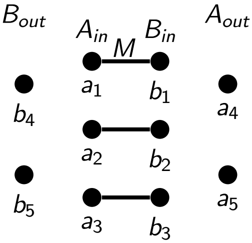
Strategy: Bound "knowledge" about input graph after each query ("structure graph"); ensure perfect matching can be added

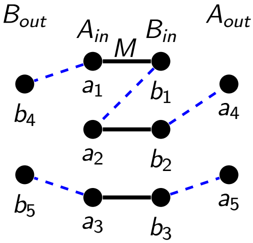
Lower Bound Construction - First Query

Strategy: Bound "knowledge" about input graph after each query ("structure graph"); ensure perfect matching can be added

First Query:

- Oracle commits to structure below and returns subset of edges M (no edges between A_{out} and B_{out})
- A perfect matching (blue edges) can be added, which implies that approximation factor is 3/5 at best after first query

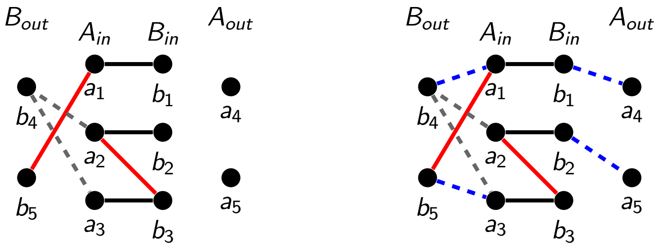




Lower Bound Construction - Second Query

Second Query:

- Information can be bounded by structure below - grey edges indicate that edges are not present in output graph
- Again, perfect matching can be added, which implies that approximation factor is 3/5 at best after second query



Lower Bound Construction - Third Query

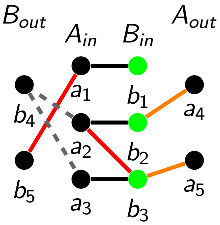
Third Query:

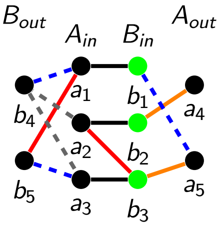
- Structure cannot easily be captured using a single “structure graph“
- Instead, case distinctions with cleverly grouping cases together

Third Query:

- Structure cannot easily be captured using a single “structure graph“
- Instead, case distinctions with cleverly grouping cases together

Example Case: Query includes $\{b_1, b_2, b_3\}$

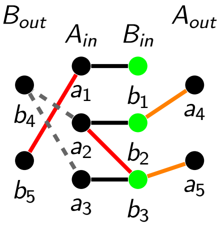


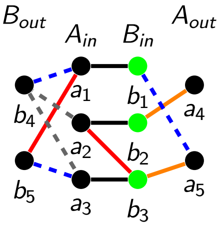


Third Query:

- Structure cannot easily be captured using a single “structure graph“
- Instead, case distinctions with cleverly grouping cases together

Example Case: Query includes $\{b_1, b_2, b_3\}$



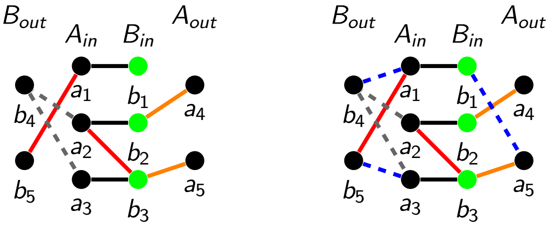


Lower Bound Construction - Third Query

Third Query:

- Structure cannot easily be captured using a single “structure graph“
- Instead, case distinctions with cleverly grouping cases together

Example Case: Query includes $\{b_1, b_2, b_3\}$



$\square$

Key Technique: Structural properties that allow eliminating cases

Open Problems and Outlook

Open Problems:

Open Problems and Outlook

Open Problems:

- Can we compute a Maximum Matching in $o(n^2)$ rounds?

Open Problems and Outlook

Open Problems:

- Can we compute a Maximum Matching in $o(n^2)$ rounds?
- Can we prove that $\Omega(1/\epsilon^2)$ rounds are required for computing a $(1-\epsilon)$-approximation?

Open Problems and Outlook

Open Problems:

- Can we compute a Maximum Matching in $o(n^2)$ rounds?
- Can we prove that $\Omega(1/\epsilon^2)$ rounds are required for computing a $(1-\epsilon)$-approximation?

Outlook:

Open Problems and Outlook

Open Problems:

- Can we compute a Maximum Matching in $o(n^2)$ rounds?
- Can we prove that $\Omega(1/\epsilon^2)$ rounds are required for computing a $(1-\epsilon)$-approximation?

Outlook:

- Extensions: Edge queries instead of vertex queries

Open Problems and Outlook

Open Problems:

- Can we compute a Maximum Matching in $o(n^2)$ rounds?
- Can we prove that $\Omega(1/\epsilon^2)$ rounds are required for computing a $(1-\epsilon)$-approximation?

Outlook:

- Extensions: Edge queries instead of vertex queries
- Randomization?

Thank you for your attention.