Online Matroid Intersection: Beating Half for Random Arrival

Sahil Singla (ssingla@cmu.edu)
Guru Prashanth Guruganesh (ggurugan@cs.cmu.edu)

Carnegie Mellon University

26th June, 2017

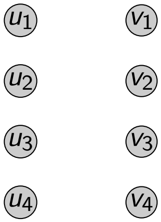
Outline

Edge arrival

- Bipartite graph: Intersection of **partition matroids**

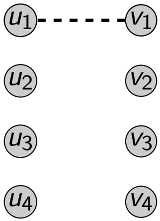
Edge arrival

- Bipartite graph: Intersection of **partition matroids**



Edge arrival

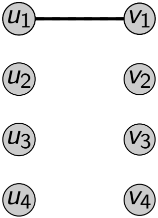
- Bipartite graph: Intersection of **partition matroids**



- Immediately & Irrevocably: Maximize size of **matching**

Edge arrival

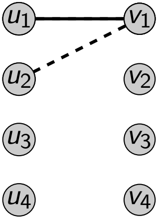
- Bipartite graph: Intersection of **partition matroids**



- Immediately & Irrevocably: Maximize size of **matching**

Edge arrival

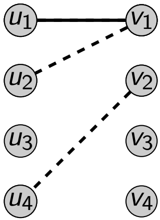
- Bipartite graph: Intersection of **partition matroids**



- Immediately & Irrevocably: Maximize size of **matching**

Edge arrival

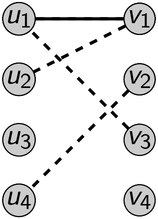
- Bipartite graph: Intersection of **partition matroids**



- Immediately & Irrevocably: Maximize size of **matching**

Edge arrival

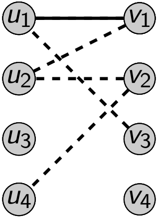
- Bipartite graph: Intersection of **partition matroids**



- Immediately & Irrevocably: Maximize size of **matching**

Edge arrival

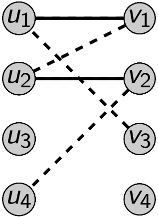
- Bipartite graph: Intersection of **partition matroids**



- Immediately & Irrevocably: Maximize size of **matching**

Edge arrival

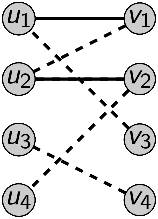
- Bipartite graph: Intersection of **partition matroids**



- Immediately & Irrevocably: Maximize size of **matching**

Edge arrival

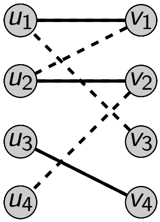
- Bipartite graph: Intersection of **partition matroids**



- Immediately & Irrevocably: Maximize size of **matching**

Edge arrival

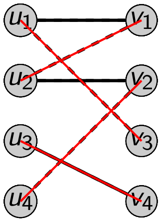
- Bipartite graph: Intersection of **partition matroids**



- Immediately & Irrevocably: Maximize size of **matching**

Edge arrival

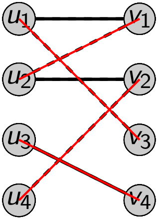
- Bipartite graph: Intersection of **partition matroids**



- Immediately & Irrevocably: Maximize size of **matching**

Edge arrival

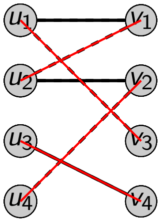
- Bipartite graph: Intersection of **partition matroids**



- Immediately & Irrevocably: Maximize size of **matching**
- GREEDY (pick an edge if possible): maximal matching

Edge arrival

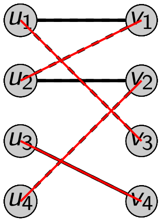
- Bipartite graph: Intersection of **partition matroids**



- Immediately & Irrevocably: Maximize size of **matching**
- GREEDY (pick an edge if possible): maximal matching
  $\frac{1}{2} \leq \frac{ALG}{OPT}$: **Competitive Ratio**

Edge arrival

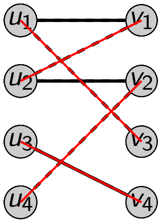
- Bipartite graph: Intersection of **partition matroids**



- Immediately & Irrevocably: Maximize size of **matching**
- GREEDY (pick an edge if possible): maximal matching
  $\frac{1}{2} \leq \frac{ALG}{OPT}$: **Competitive Ratio**
- Better algo possible?

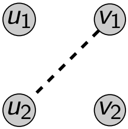
Edge arrival

- Bipartite graph: Intersection of **partition matroids**



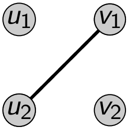
- Immediately & Irrevocably: Maximize size of **matching**
- GREEDY (pick an edge if possible): maximal matching
  $\frac{1}{2} \leq \frac{ALG}{OPT}$: **Competitive Ratio**
- Better algo possible? Adversarial/Random arrival

The Z graph



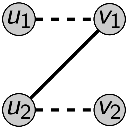
Q. Should we pick the first edge?

The Z graph



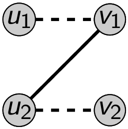
Q. Should we pick the first edge?

The Z graph



Q. Should we pick the first edge?

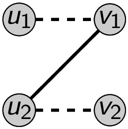
The Z graph



Q. Should we pick the first edge?

- Best **deterministic** is $\frac{1}{2}$-competitive (adversarial arrival)

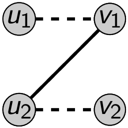
The Z graph



Q. Should we pick the first edge?

- Best **deterministic** is $\frac{1}{2}$-competitive (adversarial arrival)
- Select w.p. $\frac{2}{3}$. Gets $\frac{4}{3}$ edges in expectation!

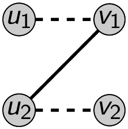
The Z graph



Q. Should we pick the first edge?

- Best **deterministic** is $\frac{1}{2}$-competitive (adversarial arrival)
- Select w.p. $\frac{2}{3}$. Gets $\frac{4}{3}$ edges in expectation!
- **Randomization** adds power: $\frac{\mathbb{E}[ALG]}{OPT}$ **Competitive Ratio**

The Z graph



Q. Should we pick the first edge?

- Best **deterministic** is $\frac{1}{2}$-competitive (adversarial arrival)
- Select w.p. $\frac{2}{3}$. Gets $\frac{4}{3}$ edges in expectation!
- **Randomization** adds power: $\frac{\mathbb{E}[ALG]}{OPT}$ **Competitive Ratio**
- Now, is better than $\frac{1}{2}$ possible?

Online Matroid Intersection

- Two **unknown** matroids $\mathcal{M}_1 = (E, \mathcal{I}_1)$ and $\mathcal{M}_2 = (E, \mathcal{I}_2)$
- Elements revealed one-by-one: Adversarial/Random arrival
- **Matroids oracles** only on the **revealed** elements
- Immediately & Irrevocably decide

Online Matroid Intersection

- Two **unknown** matroids $\mathcal{M}_1 = (E, \mathcal{I}_1)$ and $\mathcal{M}_2 = (E, \mathcal{I}_2)$
- Elements revealed one-by-one: Adversarial/Random arrival
- **Matroids oracles** only on the **revealed** elements
- Immediately & Irrevocably decide
- GREEDY (pick an element if possible) is $\frac{1}{2}$ competitive

Online Matroid Intersection

- Two **unknown** matroids $\mathcal{M}_1 = (E, \mathcal{I}_1)$ and $\mathcal{M}_2 = (E, \mathcal{I}_2)$
- Elements revealed one-by-one: Adversarial/Random arrival
- **Matroids oracles** only on the **revealed** elements
- Immediately & Irrevocably decide
- GREEDY (pick an element if possible) is $\frac{1}{2}$ competitive
- Better algo possible?

Online Matroid Intersection

- Two **unknown** matroids $\mathcal{M}_1 = (E, \mathcal{I}_1)$ and $\mathcal{M}_2 = (E, \mathcal{I}_2)$
- Elements revealed one-by-one: Adversarial/Random arrival
- **Matroids oracles** only on the **revealed** elements
- Immediately & Irrevocably decide
- GREEDY (pick an element if possible) is $\frac{1}{2}$ competitive
- Better algo possible?

Theorem

There exists a $(\frac{1}{2} + \epsilon)$-competitive algorithm when the elements are revealed in a **random order**, where $\epsilon > 10^{-5}$.

Outline

Comparison to Vertex Arrival

- **Adversarial arrival (KVV algo.[1]):** $1 - \frac{1}{e} \approx 0.63$
  (a) Give a random rank to $\{u_1, u_2, \ldots, u_n\}$
  (b) Match v_i to lowest available u_j

[1] Karp-Vazirani-Vazirani STOC '90
[2] Mahdian-Yan STOC '11
[3] Wajc, Unpublished

Comparison to Vertex Arrival

- **Adversarial arrival (KVV algo.[1]):** $1 - \frac{1}{e} \approx 0.63$
  (a) Give a random rank to $\{u_1, u_2, \ldots, u_n\}$
  (b) Match v_i to lowest available u_j

- **Random arrival (MY algo.[2])**: > 0.69

[1] Karp-Vazirani-Vazirani STOC '90
[2] Mahdian-Yan STOC '11
[3] Wajc, Unpublished

Comparison to Vertex Arrival

- **Adversarial arrival (KVV algo.[1]):** $1 - \frac{1}{e} \approx 0.63$
  - (a) Give a random rank to $\{u_1, u_2, \ldots, u_n\}$
  - (b) Match v_i to lowest available u_j
- **Random arrival (MY algo.[2])**: > 0.69

	Vertex arriv	**Edge arriv**
Random	> 0.69	
Adversarial	≈ 0.63	3

[1] Karp-Vazirani-Vazirani STOC '90
[2] Mahdian-Yan STOC '11
[3] Wajc, Unpublished

Comparison to Vertex Arrival

- **Adversarial arrival (KVV algo.[1]):** $1 - \frac{1}{e} \approx 0.63$
  (a) Give a random rank to $\{u_1, u_2, \ldots, u_n\}$
  (b) Match v_i to lowest available u_j
- **Random arrival (MY algo.[2])**: > 0.69

	Vertex arriv	**Edge arriv**
Random	> 0.69	
Adversarial	≈ 0.63	$\geq \frac{1}{2}$ & < 0.572[3]

[1] Karp-Vazirani-Vazirani STOC '90
[2] Mahdian-Yan STOC '11
[3] Wajc, Unpublished

Comparison to Vertex Arrival

- **Adversarial arrival (KVV algo.[1]):** $1 - \frac{1}{e} \approx 0.63$
  (a) Give a random rank to $\{u_1, u_2, \ldots, u_n\}$
  (b) Match v_i to lowest available u_j

- **Random arrival (MY algo.[2])**: > 0.69

	Vertex arriv	**Edge arriv**
Random	> 0.69	$> \frac{1}{2} + \epsilon$ & < 0.822
Adversarial	≈ 0.63	$\geq \frac{1}{2}$ & < 0.572[3]

[1]Karp-Vazirani-Vazirani STOC '90
[2]Mahdian-Yan STOC '11
[3]Wajc, Unpublished

Faster Algorithms

Offline Algorithms

[4] Hopcroft-Karp SICOMP'73

[5] Chekuri-Quanrud, SODA'16 and Huang et al., SODA'16

[6] Harvey, SODA'08

Faster Algorithms

Offline Algorithms

- Linear time $(1-\epsilon)$-approx max cardinality matching[4]

[4] Hopcroft-Karp SICOMP'73

[5] Chekuri-Quanrud, SODA'16 and Huang et al., SODA'16

[6] Harvey, SODA'08

Faster Algorithms

Offline Algorithms

- Linear time $(1-\epsilon)$-approx max cardinality matching[4]
- Recent works give **quadratic time** $(1-\epsilon)$-approx algos for max-weight matroid intersection[5]

---

[4] Hopcroft-Karp SICOMP'73

[5] Chekuri-Quanrud, SODA'16 and Huang et al., SODA'16

[6] Harvey, SODA'08

Faster Algorithms

Offline Algorithms

- Linear time $(1-\epsilon)$-approx max cardinality matching[4]
- Recent works give **quadratic time** $(1-\epsilon)$-approx algos for max-weight matroid intersection[5]
- Our algorithm gives first **linear time** $(1/2+\epsilon)$-approx algo for max-cardinality matroid intersection

[4] Hopcroft-Karp SICOMP'73
[5] Chekuri-Quanrud, SODA'16 and Huang et al., SODA'16
[6] Harvey, SODA'08

Faster Algorithms

Offline Algorithms

- Linear time $(1-\epsilon)$-approx max cardinality matching[4]
- Recent works give **quadratic time** $(1-\epsilon)$-approx algos for max-weight matroid intersection[5]
- Our algorithm gives first **linear time** $(1/2+\epsilon)$-approx algo for max-cardinality matroid intersection
- Even for exact matroid intersection, only linear time lower bounds known[6]

---

[4] Hopcroft-Karp SICOMP'73
[5] Chekuri-Quanrud, SODA'16 and Huang et al., SODA'16
[6] Harvey, SODA'08

Other Edge Arrival Models

- **Edge Weighted Bipartite Matching**
  - (a) Maximize weight of matching
  - (b) No constant approx possible for adversarial arrival
  - (c) For random arrival, constant approx possible[7]

[7] Korula-Pal, ICALP'09 and Kesselheim et al., ESA'13
[8] Paz-Schwartzman, SODA'17
[9] Konrad et al., APPROX'12

Other Edge Arrival Models

- **Edge Weighted Bipartite Matching**
  (a) Maximize weight of matching
  (b) No constant approx possible for adversarial arrival
  (c) For random arrival, constant approx possible[7]
- **Semi-Streaming Models**
  (a) Decisions for $\tilde{O}(n)$ edges can be postponed
  (b) For edge-weighted, $1/2 - \epsilon$ recently shown[8]
  (c) For unweighted, $1/2 + \epsilon$ known when edges arrive randomly[9]

[7] Korula-Pal, ICALP'09 and Kesselheim et al., ESA'13
[8] Paz-Schwartzman, SODA'17
[9] Konrad et al., APPROX'12

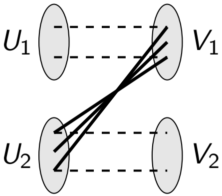
Outline

GREEDY algorithm – random edge arrival

- GREEDY **algorithm**: Pick the edge if you can

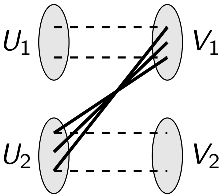
GREEDY algorithm – random edge arrival

- GREEDY **algorithm**: Pick the edge if you can
- **Thick-Z** graph:



GREEDY algorithm – random edge arrival

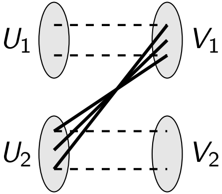
- GREEDY **algorithm**: Pick the edge if you can
- **Thick-Z** graph:



- Only $\frac{1}{2} + o(1)$ approx – **bad graph**

GREEDY algorithm – random edge arrival

- GREEDY **algorithm**: Pick the edge if you can
- **Thick-Z** graph:



- Only $\frac{1}{2} + o(1)$ approx – **bad graph**
- Regular graphs > 0.63 approx

Can assume GREEDY is bad

- Design **ALG** that gives $\frac{1}{2} + \epsilon$ for '**bad**' graphs

Can assume GREEDY is bad

- Design **ALG** that gives $\frac{1}{2} + \epsilon$ for '**bad**' graphs

	Good graphs	**Bad Graphs**
GREEDY	$\geq \frac{1}{2} + \epsilon$ (= 50.1%)	$\geq \frac{1}{2}$
ALG	≥ 0	$\geq \frac{1}{2} + \epsilon$ (= 50.1%)

Can assume GREEDY is bad

- Design **ALG** that gives $\frac{1}{2} + \epsilon$ for '**bad**' graphs

	Good graphs	**Bad Graphs**
GREEDY	$\geq \frac{1}{2} + \epsilon \quad (= 50.1\%)$	$\geq \frac{1}{2}$
ALG	≥ 0	$\geq \frac{1}{2} + \epsilon \quad (= 50.1\%)$

- Run GREEDY w.p. $1 - \epsilon$ $(= 99.9\%)$
  and **ALG** w.p. ϵ $(= 0.1\%)$

Can assume GREEDY is bad

- Design **ALG** that gives $\frac{1}{2} + \epsilon$ for '**bad**' graphs

	Good graphs	**Bad Graphs**
GREEDY	$\geq \frac{1}{2} + \epsilon$ (= 50.1%)	$\geq \frac{1}{2}$
ALG	≥ 0	$\geq \frac{1}{2} + \epsilon$ (= 50.1%)

- Run GREEDY w.p. $1 - \epsilon$ (= 99.9%)
  and **ALG** w.p. ϵ (= 0.1%)
- Now, $\mathbb{E}[Good] \geq (1/2 + \epsilon)(1 - \epsilon) + 0 = 1/2 + \epsilon/2 - \epsilon^2$
  and $\mathbb{E}[Bad] \geq 1/2(1 - \epsilon) + \epsilon(1/2 + \epsilon) = 1/2 + \epsilon^2$.

Prior work

- **Hastiness Lemma [Konrad-Magniez-Mathieu[10]]**:
  If GREEDY is **bad** then whatever it picks, it picks quickly

[10] *Maximum matching in semi-streaming with few passes.*, APPROX '12

Prior work

- **Hastiness Lemma [Konrad-Magniez-Mathieu[10]]**:
  If GREEDY is **bad** then whatever it picks, it picks quickly

$$\text{If } \mathbb{E}[\textsc{Greedy}\ (100\%)] < \frac{1}{2} + \epsilon \quad (50.1\%)$$

$$\text{then } \mathbb{E}[\textsc{Greedy}\ (10\%)] \geq \frac{1}{2} - 10\epsilon \quad (49\%)$$

[10] *Maximum matching in semi-streaming with few passes.*, APPROX '12

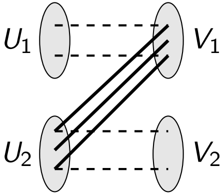
Proof idea

Assume we know GREEDY is bad

Proof idea

Assume we know GREEDY is bad

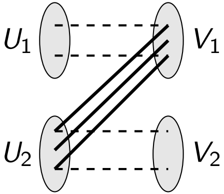
- Suppose GREEDY for first 10% edges



Proof idea

Assume we know GREEDY is bad

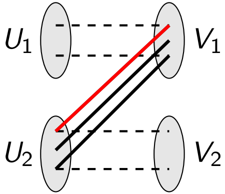
- Suppose GREEDY for first 10% edges – close to half



Proof idea

Assume we know GREEDY is bad

- Suppose GREEDY for first 10% edges – close to half

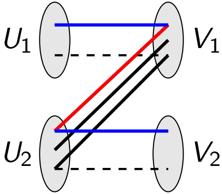


- Would like to **'mark'** some edges

Proof idea

Assume we know GREEDY is bad

- Suppose GREEDY for first 10% edges – close to half

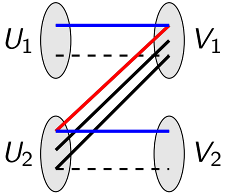


- Would like to **'mark'** some edges and '**augment**' them later

Proof idea

Assume we know GREEDY is bad

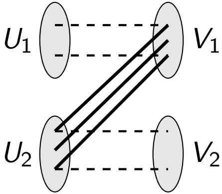
- Suppose GREEDY for first 10% edges – close to half



- Would like to **'mark'** some edges and '**augment**' them later
- What edges are augmentable?

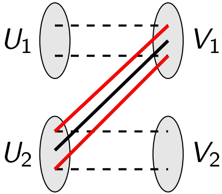
Two Phase Algorithm **ALG**

(a) GREEDY for 10% edges



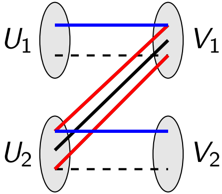
Two Phase Algorithm **ALG**

(a) GREEDY for 10% edges – but **randomly mark** 20%



Two Phase Algorithm **ALG**

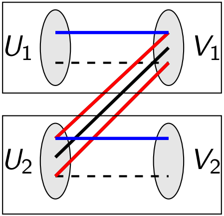
(a) GREEDY for 10% edges – but **randomly mark** 20%



(b) Try **augmenting** marked

Two Phase Algorithm **ALG**

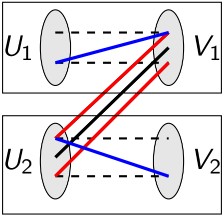
(a) GREEDY for 10% edges – but **randomly mark** 20%



(b) Try **augmenting** marked – For next 90% edges
Run GREEDY (U_1, V_1) and GREEDY (U_2, V_2)

Two Phase Algorithm **ALG**

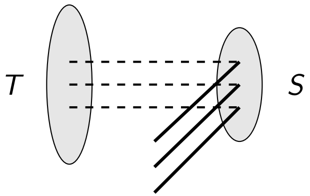
(a) GREEDY for 10% edges – but **randomly mark** 20%



(b) Try **augmenting** marked – For next 90% edges
Run GREEDY (U_1, V_1) and GREEDY (U_2, V_2)

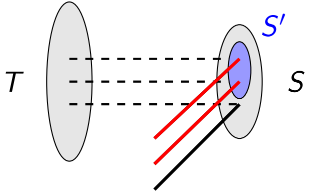
- Augmentations kill each other?

Random sampling



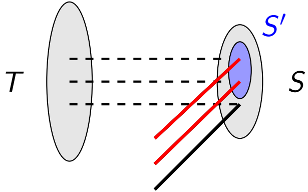
- Bip. graph (T, S) with S-perfect matching

Random sampling



- Bip. graph (T, S) with S-perfect matching
- $S' \subseteq S$ with sampling prob 0.2

Random sampling



- Bip. graph (T, S) with S-perfect matching
- $S' \subseteq S$ with sampling prob 0.2
- $\mathbb{E}[\textsc{Greedy}\ (T, S')]$: Better than $\mathbb{E}[|S'|]\ (\frac{1}{2})$?

Sampling Lemma

Q. $\mathbb{E}[\textsc{Greedy}\ (T, S')]$: Better than $\mathbb{E}[|S'|]\left(\frac{1}{2}\right)$?

Sampling Lemma

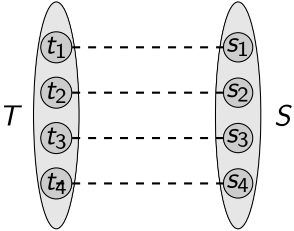
Q. $\mathbb{E}[\textsc{Greedy}\ (T, S')]$: Better than $\mathbb{E}[|S'|]\left(\frac{1}{2}\right)$?

A. Yes, $\geq \mathbb{E}[|S'|]\left(\frac{1}{1+0.2}\right)$

Sampling Lemma

Q. $\mathbb{E}[\text{GREEDY}\,(T, S')]$: Better than $\mathbb{E}[|S'|]\left(\frac{1}{2}\right)$?

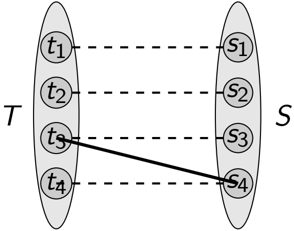
A. Yes, $\geq \mathbb{E}[|S'|]\left(\frac{1}{1+0.2}\right)$



Sampling Lemma

Q. $\mathbb{E}[\text{GREEDY}\ (T, S')]$: Better than $\mathbb{E}[|S'|]\left(\frac{1}{2}\right)$?

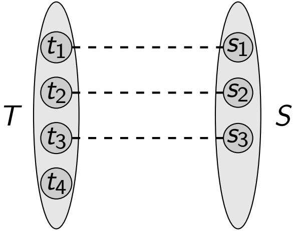
A. Yes, $\geq \mathbb{E}[|S'|]\left(\frac{1}{1+0.2}\right)$



Sampling Lemma

Q. $\mathbb{E}[\textsc{Greedy}\ (T, S')]$: Better than $\mathbb{E}[|S'|]\left(\frac{1}{2}\right)$?

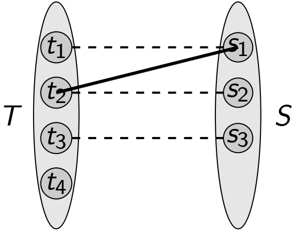
A. Yes, $\geq \mathbb{E}[|S'|]\left(\frac{1}{1+0.2}\right)$



Sampling Lemma

Q. $\mathbb{E}[\textsc{Greedy}\,(T, S')]$: Better than $\mathbb{E}[|S'|]\left(\frac{1}{2}\right)$?

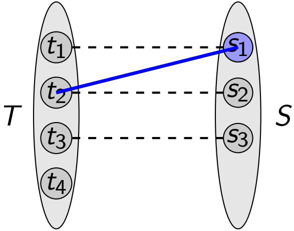
A. Yes, $\geq \mathbb{E}[|S'|]\left(\frac{1}{1+0.2}\right)$



Sampling Lemma

Q. $\mathbb{E}[\text{GREEDY}\ (T, S')]$: Better than $\mathbb{E}[|S'|]\left(\frac{1}{2}\right)$?

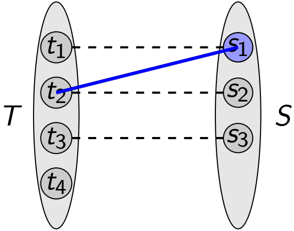
A. Yes, $\geq \mathbb{E}[|S'|]\left(\frac{1}{1+0.2}\right)$



Sampling Lemma

Q. $\mathbb{E}[\text{GREEDY}\ (T, S')]$: Better than $\mathbb{E}[|S'|]\left(\frac{1}{2}\right)$?

A. Yes, $\geq \mathbb{E}[|S'|]\left(\frac{1}{1+0.2}\right)$

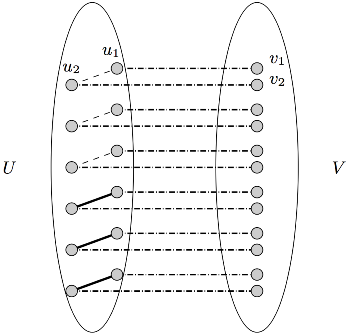


- Note s_2 marked w.p. only 0.2

Outline

General Matching

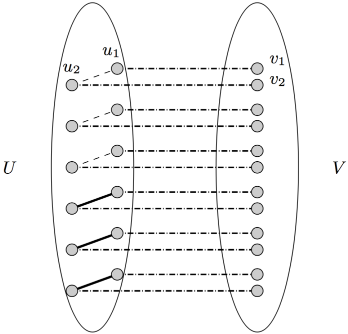
Assume GREEDY is bad



- U denotes vertices matched by GREEDY (in Phase (a))

General Matching

Assume GREEDY is bad



- U denotes vertices matched by GREEDY (in Phase (a))
- Reduces to bipartite matching problem

Matroid Intersection

- Assume GREEDY is bad

Matroid Intersection

- Assume GREEDY is bad
- Extend **Hastiness Lemma**

Matroid Intersection

- Assume GREEDY is bad
- Extend **Hastiness Lemma**
- Run GREEDY with Marking in Phase (a):
  let T_f be the GREEDY and S be the picked elements

Matroid Intersection

- Assume GREEDY is bad
- Extend **Hastiness Lemma**
- Run GREEDY with Marking in Phase (a):
  let T_f be the GREEDY and S be the picked elements
- In Phase (b):
  - Consider e only if in **span of exactly one matroid**, say $span_1(T_f)$
  - Pick only if e **independent w.r.t.** S **in** $\mathcal{M}_1$ and **w.r.t.** T_f **in** $\mathcal{M}_2$, along with the **newly picked** elements.

Matroid Intersection

- Assume GREEDY is bad
- Extend **Hastiness Lemma**
- Run GREEDY with Marking in Phase (a):
  let T_f be the GREEDY and S be the picked elements
- In Phase (b):
  - Consider e only if in **span of exactly one matroid**, say $span_1(T_f)$
  - Pick only if e **independent w.r.t.** S **in** $\mathcal{M}_1$ and **w.r.t.** T_f **in** $\mathcal{M}_2$, along with the **newly picked** elements.
- Extend **Sampling Lemma**

Outline

Open Problems

Question 1

Is there a linear time $(1-\epsilon)$-approximation algorithm for offline matroid intersection?

Open Problems

Question 1

Is there a linear time $(1 - \epsilon)$-approximation algorithm for offline matroid intersection?

Question 2

Can we beat half for adversarial edge arrival?

Open Problems

Question 1

Is there a linear time $(1-\epsilon)$-approximation algorithm for offline matroid intersection?

Question 2

Can we beat half for adversarial edge arrival?

Question 3

For OMI, can we "significantly" improve the $(\frac{1}{2}+\epsilon)$-competitive ratio?

Conclusion

- **Random edge arrival**
  - Showed $(\frac{1}{2} + \epsilon)$-approx for bipartite graphs
  - Use Hastiness Lemma and Sampling Lemma
  - Cannot do better than 0.822

Conclusion

- **Random edge arrival**
  - Showed $(\frac{1}{2} + \epsilon)$-approx for bipartite graphs
  - Use Hastiness Lemma and Sampling Lemma
  - Cannot do better than 0.822
- **Extensions**
  - General Graphs
  - Online Matroid Intersection

Conclusion

- **Random edge arrival**
  - Showed $(\frac{1}{2} + \epsilon)$-approx for bipartite graphs
  - Use Hastiness Lemma and Sampling Lemma
  - Cannot do better than 0.822
- **Extensions**
  - General Graphs
  - Online Matroid Intersection
- **Open problems**
  - Linear time $(1 - \epsilon)$-approx matroid intersection?
  - Can we beat half for adversarial edge arrival?

Conclusion

- **Random edge arrival**
  - Showed $(\frac{1}{2} + \epsilon)$-approx for bipartite graphs
  - Use Hastiness Lemma and Sampling Lemma
  - Cannot do better than 0.822
- **Extensions**
  - General Graphs
  - Online Matroid Intersection
- **Open problems**
  - Linear time $(1 - \epsilon)$-approx matroid intersection?
  - Can we beat half for adversarial edge arrival?

QUESTIONS?