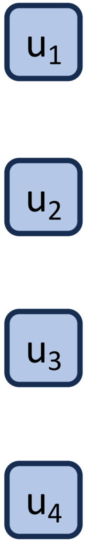
Online bipartite matching with imperfect advice

Davin Choo, Themis **G**ouleakis, Chun Kai **L**ing, Arnab **B**hattacharyya



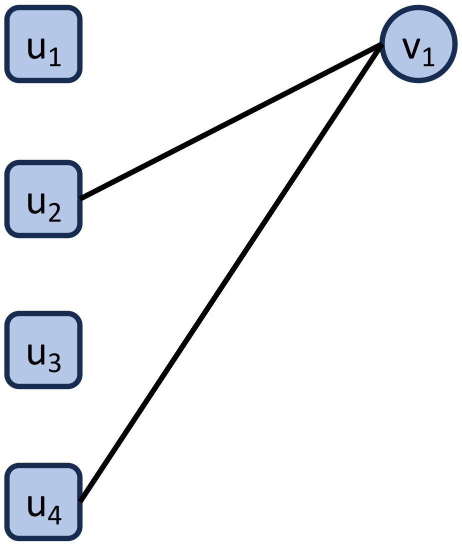

Online bipartite matching

- Offline set $\mathrm{U} = \{\mathrm{u}_1, \ldots, \mathrm{u}_\mathrm{n}\}$ fixed and known
- Online set $\mathrm{V} = \{\mathrm{v}_1, \ldots, \mathrm{v}_\mathrm{n}\}$ arrive one by one



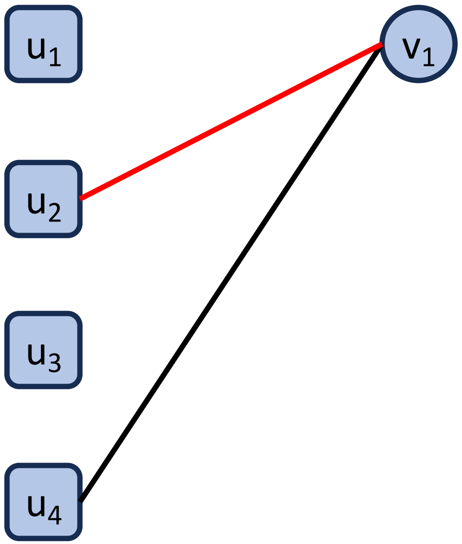
Online bipartite matching

- Offline set $U = \{u_1, \dots, u_n\}$ fixed and known
- Online set $V = \{v_1, \dots, v_n\}$ arrive one by one
- When an online vertex v_i arrives
  - Its neighbors $N(v_i)$ are revealed
  - We have a make an irrevocable decision whether, how, to match v_i to something in $N(v_i)$



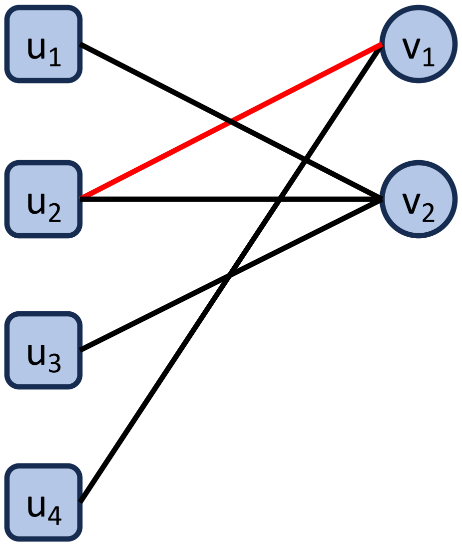
Online bipartite matching

- Offline set $\mathrm{U} = \{\mathrm{u}_1, \ldots, \mathrm{u}_\mathrm{n}\}$ fixed and known
- Online set $\mathrm{V} = \{\mathrm{v}_1, \ldots, \mathrm{v}_\mathrm{n}\}$ arrive one by one
- When an online vertex v_i arrives
  - Its neighbors $\mathrm{N}(\mathrm{v}_\mathrm{i})$ are revealed
  - We have a make an irrevocable decision whether, how, to match v_i to something in $\mathrm{N}(\mathrm{v}_\mathrm{i})$



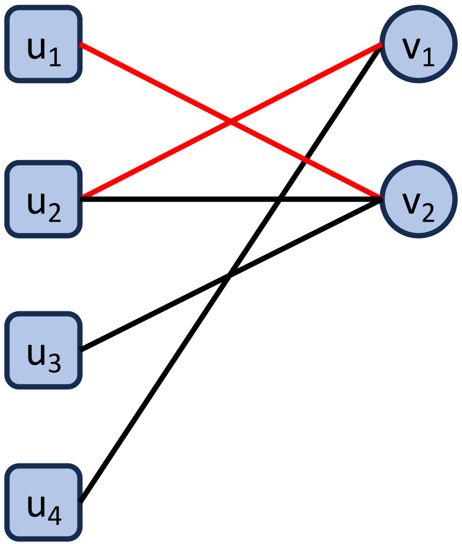
Online bipartite matching

- Offline set $U = \{u_1, \ldots, u_n\}$ fixed and known
- Online set $V = \{v_1, \ldots, v_n\}$ arrive one by one
- When an online vertex v_i arrives
  - Its neighbors $N(v_i)$ are revealed
  - We have a make an irrevocable decision whether, how, to match v_i to something in $N(v_i)$



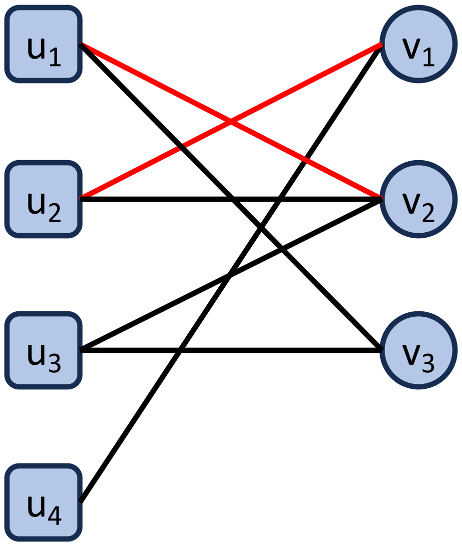
Online bipartite matching

- Offline set $U = \{u_1, \dots, u_n\}$ fixed and known
- Online set $V = \{v_1, \dots, v_n\}$ arrive one by one
- When an online vertex v_i arrives
  - Its neighbors $N(v_i)$ are revealed
  - We have a make an irrevocable decision whether, how, to match v_i to something in $N(v_i)$



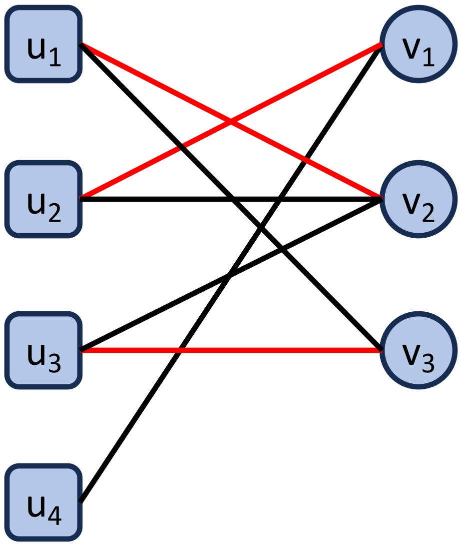
Online bipartite matching

- Offline set $U = \{u_1, \dots, u_n\}$ fixed and known
- Online set $V = \{v_1, \dots, v_n\}$ arrive one by one
- When an online vertex v_i arrives
  - Its neighbors $N(v_i)$ are revealed
  - We have a make an irrevocable decision whether, how, to match v_i to something in $N(v_i)$



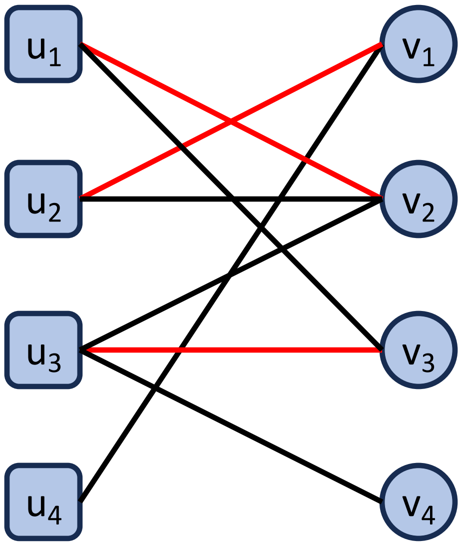
Online bipartite matching

- Offline set $U = \{u_1, \ldots, u_n\}$ fixed and known
- Online set $V = \{v_1, \ldots, v_n\}$ arrive one by one
- When an online vertex v_i arrives
  - Its neighbors $N(v_i)$ are revealed
  - We have a make an irrevocable decision whether, how, to match v_i to something in $N(v_i)$



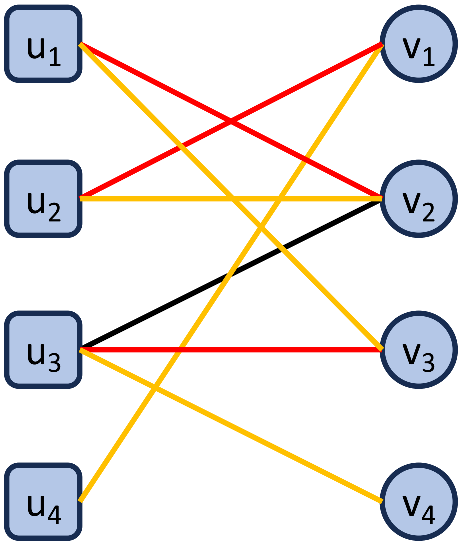
Online bipartite matching

- Offline set $U = \{u_1, \dots, u_n\}$ fixed and known
- Online set $V = \{v_1, \dots, v_n\}$ arrive one by one
- When an online vertex v_i arrives
  - Its neighbors $N(v_i)$ are revealed
  - We have a make an irrevocable decision whether, how, to match v_i to something in $N(v_i)$



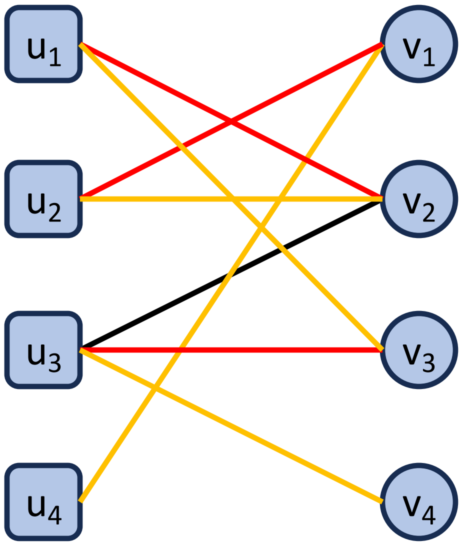
Online bipartite matching

- Offline set $U = \{u_1, \ldots, u_n\}$ fixed and known
- Online set $V = \{v_1, \ldots, v_n\}$ arrive one by one
- When an online vertex v_i arrives
  - Its neighbors $N(v_i)$ are revealed
  - We have a make an irrevocable decision whether, how, to match v_i to something in $N(v_i)$
- Final offline graph $G^* = (U \cup V, E)$
  - $E = N(v_1) \cup \cdots \cup N(v_n)$
  - Maximum matching $M^* \subseteq E$ of size $|M^*| = n^* \leq n$



Online bipartite matching

- Offline set $U = \{u_1, \dots, u_n\}$ fixed and known
- Online set $V = \{v_1, \dots, v_n\}$ arrive one by one
- Final offline graph $G^* = (U \cup V, E)$
  - $E = N(v_1) \cup \cdots \cup N(v_n)$
  - Maximum matching $M^* \subseteq E$ of size $|M^*| = n^* \leq n$



Here, the ratio is $3/4$

Goal of online bipartite matching problem

Produce a matching M such that the resulting competitive ratio $\frac{|M|}{|M^*|}$ is **maximized**

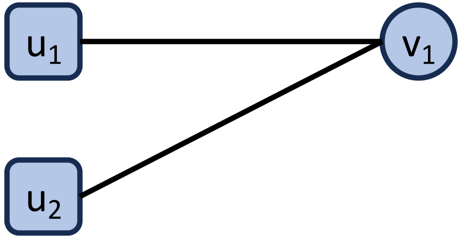
For this talk, let's treat $n^* = n$

What is known?

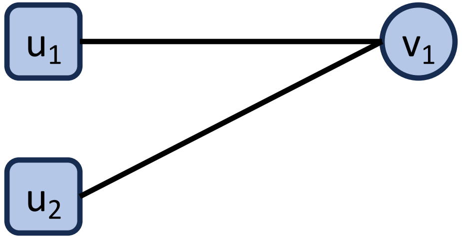
- Any reasonable greedy algorithm has competitive ratio $\geq 1/2$
  - Size of maxi<u>mal</u> matching is at least half of size of maxi<u>mum</u> matching

What is known?

- Any reasonable greedy algorithm has competitive ratio $\geq 1/2$
  - Size of maxi<u>mal</u> matching is at least half of size of maxi<u>mum</u> matching
- Why is online bipartite matching hard?
  - Maximum bipartite matching is poly time computable…
  - But we don't know the future in the online setting!

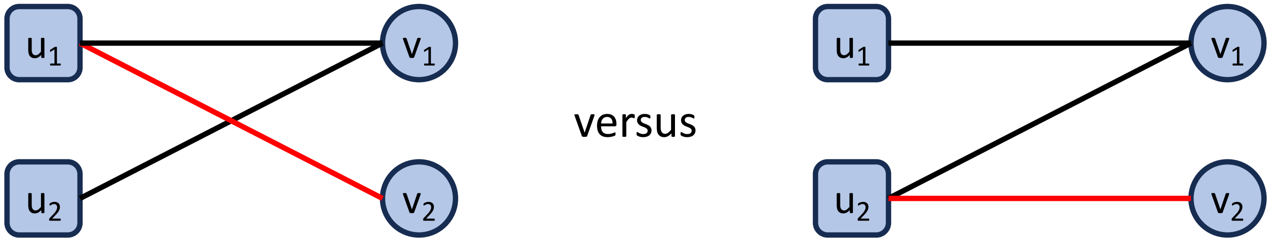


versus



What is known?

- Any reasonable greedy algorithm has competitive ratio $\geq 1/2$
  - Size of maxi<u>mal</u> matching is at least half of size of maxi<u>mum</u> matching
- Why is online bipartite matching hard?
  - Maximum bipartite matching is poly time computable…
  - But we don't know the future in the online setting!



What is known?

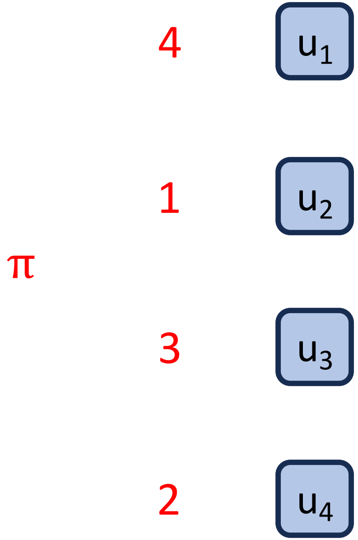
$$\min_{G} \min_{V'\text{s arrival sequence}} \frac{\text{(Expected) number of matches}}{n^*}$$

	(Expected) Competitive ratio
Deterministic algorithm	$\frac{1}{2}$ ← Greedy
Deterministic hardness	$\frac{1}{2}$
Randomized algorithm	$1-\frac{1}{e}$ [KVV90] ← Ranking
Randomized hardness	$1-\frac{1}{e}+o(1)$ [KVV90]

- The Ranking algorithm [KVV90]
  - Pick a random permutation π over the offline vertices U
  - When vertex v_i arrive with $N(v_i)$, match v_i to the smallest indexed (with respect to π) unmatched neighbor

[KVV90] Richard M. Karp, Umesh V. Vazirani, Vijay V. Vazirani. *An optimal algorithm for on-line bipartite matching*. Symposium on Theory of Computing (STOC), 1990

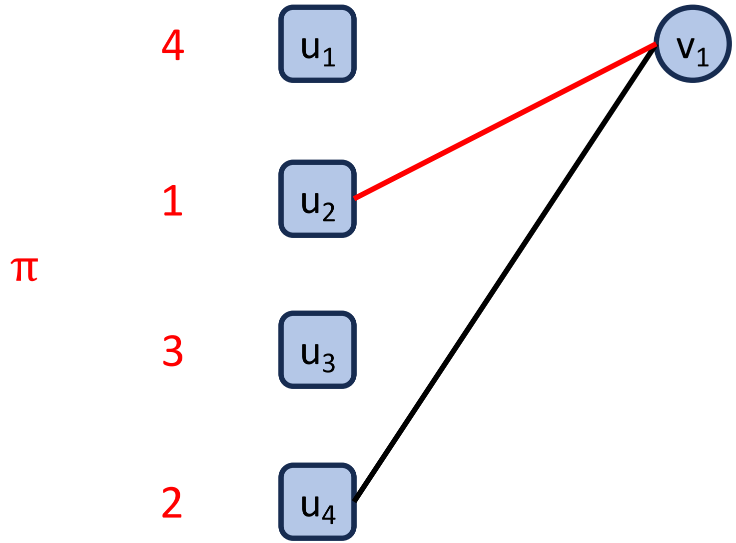
What is known?



- The Ranking algorithm [KVV90]
  - Pick a random permutation π over the offline vertices U
  - When vertex v_i arrive with $N(v_i)$, match v_i to the smallest indexed (with respect to π) unmatched neighbor

[KVV90] Richard M. Karp, Umesh V. Vazirani, Vijay V. Vazirani. *An optimal algorithm for on-line bipartite matching*. Symposium on Theory of Computing (STOC), 1990

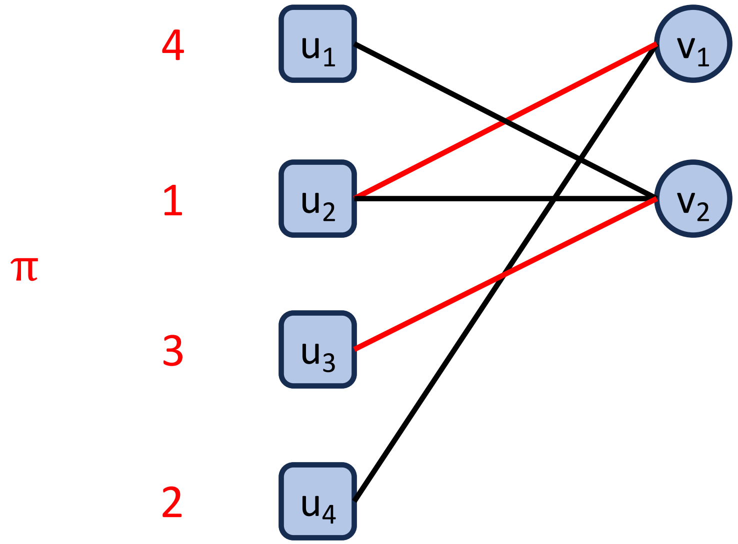
What is known?



- The Ranking algorithm [KVV90]
  - Pick a random permutation π over the offline vertices U
  - When vertex v_i arrive with $N(v_i)$, match v_i to the smallest indexed (with respect to π) unmatched neighbor

[KVV90] Richard M. Karp, Umesh V. Vazirani, Vijay V. Vazirani. *An optimal algorithm for on-line bipartite matching*. Symposium on Theory of Computing (STOC), 1990

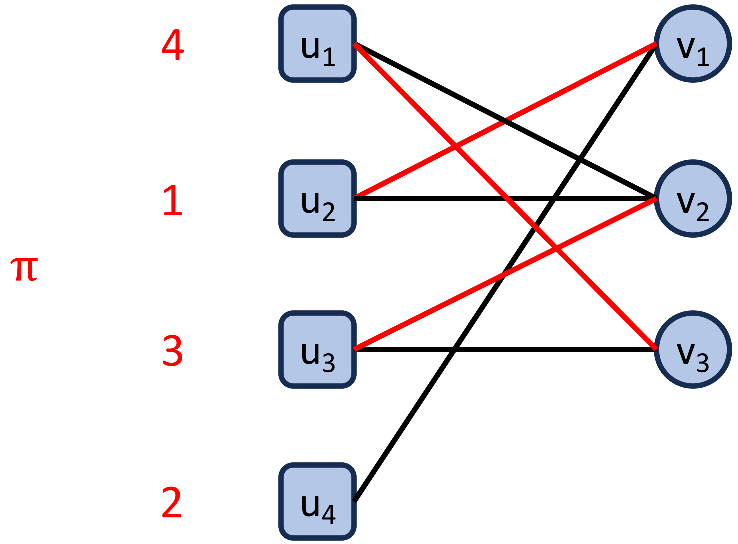
What is known?



- The Ranking algorithm [KVV90]
  - Pick a random permutation π over the offline vertices U
  - When vertex v_i arrive with $N(v_i)$, match v_i to the smallest indexed (with respect to π) unmatched neighbor

[KVV90] Richard M. Karp, Umesh V. Vazirani, Vijay V. Vazirani. *An optimal algorithm for on-line bipartite matching*. Symposium on Theory of Computing (STOC), 1990

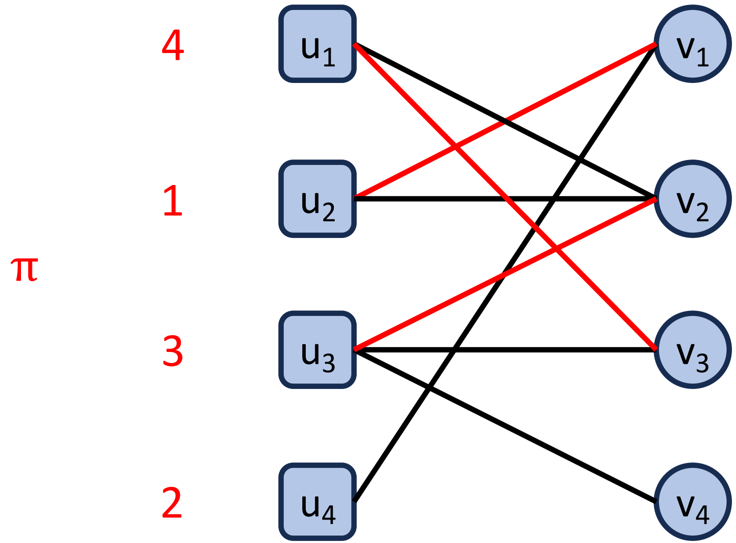
What is known?



- The Ranking algorithm [KVV90]
  - Pick a random permutation π over the offline vertices U
  - When vertex v_i arrive with $N(v_i)$, match v_i to the smallest indexed (with respect to π) unmatched neighbor

[KVV90] Richard M. Karp, Umesh V. Vazirani, Vijay V. Vazirani. *An optimal algorithm for on-line bipartite matching*. Symposium on Theory of Computing (STOC), 1990

What is known?



- The Ranking algorithm [KVV90]
  - Pick a random permutation π over the offline vertices U
  - When vertex v_i arrive with $N(v_i)$, match v_i to the smallest indexed (with respect to π) unmatched neighbor

[KVV90] Richard M. Karp, Umesh V. Vazirani, Vijay V. Vazirani. *An optimal algorithm for on-line bipartite matching*. Symposium on Theory of Computing (STOC), 1990

What if there is additional side information?

- Learning-augmented algorithms
  - Designing algorithms using advice, predictions, etc.
    - α-consistent: α-competitive with no advice error
    - β-robust: β-competitive with any advice error

A natural goal is to design an algorithm with $\alpha = 1$ while β being the best possible classically

What if there is additional side information?

- Learning-augmented algorithms
  - Designing algorithms using advice, predictions, etc.
    - α-consistent: α-competitive with no advice error
    - β-robust: β-competitive with any advice error
- Example: Binary search with advice
  - Want to find a word in an n page dictionary, say it is on page x^*
  - Classical binary search: $O(\log n)$ queries possible and worst case necessary

What if there is additional side information?

- Learning-augmented algorithms
  - Designing algorithms using advice, predictions, etc.
    - α-consistent: α-competitive with no advice error
    - β-robust: β-competitive with any advice error
- Example: Binary search with advice
  - Want to find a word in an n page dictionary, say it is on page x^*
  - Classical binary search: $O(\log n)$ queries possible and worst case necessary
  - If someone provides an advice page $\hat{x}$, $O(\log |x^* - \hat{x}|)$ queries is possible

$\hat{x}$ obtained via letter frequency tables, someone who searched a "nearby" word, or asking ChatGPT, etc…

What if there is additional side information?

- Learning-augmented algorithms
  - Designing algorithms using advice, predictions, etc.
    - α-consistent: α-competitive with no advice error
    - β-robust: β-competitive with any advice error
- Example: Binary search with advice
  - Want to find a word in an n page dictionary, say it is on page x^*
  - Classical binary search: $O(\log n)$ queries possible and worst case necessary
  - If someone provides an advice page $\hat{x}$, $O(\log |x^* - \hat{x}|)$ queries is possible
  - Here, "best possible" is directly querying to page x^*
  - So, this algorithm is 1-consistent and $O(\log n)$-robust since $|x^* - \hat{x}| \leq n$

What if there is additional side information?

Offline vertices	Online vertices	Presence of edge	Advice	Error
Advertisers	Ad slots	New ad slot fits the advertisers' requirements	Historical data	Data may have noise, bias, etc.
Food bento boxes	Conference attendee	Attendee's dietary options match the food type	Food preferences	May change mind if see a tastier option
Job opening	Hiring company	Applicant's suitability for the job role	LinkedIn qualifications	May lie about credentials

Prior attempts

- [AGKK20] Prediction on edge weights adjacent to V under an optimal offline matching
  - Random vertex arrivals and weighted edges
  - Require hyper-parameter to quantify confidence in advice, so their consistency/robustness tradeoffs are not directly comparable.

[AGKK20] Antonios Antoniadis, Themis Gouleakis, Pieter Kleer, and Pavel Kolev. *Secretary and online matching problems with machine learned advice*. Neural Information Processing Systems (NeurIPS), 2020

Prior attempts

- [AGKK20] Prediction on edge weights adjacent to V under an optimal offline matching
  - Random vertex arrivals and weighted edges
  - Require hyper-parameter to quantify confidence in advice, so their consistency/robustness tradeoffs are not directly comparable.
- [ACI22] Prediction of vertex degrees $\hat{d}(u_1), \ldots, \hat{d}(u_n)$ of the offline vertices in U
  - Adversarial arrival model
  - Optimal under the Chung-Lu-Vu random graph model [CLV03]
  - Unable to attain 1-consistency in general

[ACI22] Anders Aamand, Justin Chen, and Piotr Indyk. *(Optimal) Online Bipartite Matching with Degree Information*. Neural Information Processing Systems (NeurIPS), 2022
[CLV03] Fan Chung, Linyuan Lu, and Van Vu. *Spectra of random graphs with given expected degrees*. Proceedings of the National Academy of Sciences (PNAS), 2003

Prior attempts

- [AGKK20] Prediction on edge weights adjacent to V under an optimal offline matching
  - Random vertex arrivals and weighted edges
  - Require hyper-parameter to quantify confidence in advice, so their consistency/robustness tradeoffs are not directly comparable.
- [ACI22] Prediction of vertex degrees $\hat{d}(u_1), \ldots, \hat{d}(u_n)$ of the offline vertices in U
  - Adversarial arrival model
  - Optimal under the Chung-Lu-Vu random graph model [CLV03]
  - Unable to attain 1-consistency in general
- [JM22] Advice is a proposed matching for the first batch of arrived vertices
  - Two-staged arrival model [FNS21], where best possible robustness is ¾
  - For any $R \in [0, ¾]$, they can achieve consistency of $1 - \left(1 - \sqrt{1 - R}\right)^2$

[JM22] Billy Jin and Will Ma. *Online bipartite matching with advice: Tight robustness-consistency tradeoffs for the two-stage model*. Neural Information Processing Systems (NeurIPS), 2022

[FNS21] Yiding Feng, Rad Niazadeh, and Amin Saberi. *Two-stage stochastic matching with application to ride hailing*. Symposium on Discrete Algorithms (SODA), 2021.

Prior attempts

- [AGKK20] Prediction on edge weights adjacent to V under an optimal offline matching
  - Random vertex arrivals and weighted edges
  - Require hyper-parameter to quantify confidence in advice, so their consistency/robustness tradeoffs are not directly comparable.
- [ACI22] Prediction of vertex degrees $\hat{d}(u_1), \ldots, \hat{d}(u_n)$ of the offline vertices in U
  - Adversarial arrival model
  - Optimal under the Chung-Lu-Vu random graph model [CLV03]
  - Unable to attain 1-consistency in general
- [JM22] Advice is a proposed matching for the first batch of arrived vertices
  - Two-staged arrival model [FNS21], where best possible robustness is ¾
  - For any $R \in [0, ¾]$, they can achieve consistency of $1 - \left(1 - \sqrt{1-R}\right)^2$
- [LYR23] Augment any "expert algorithm" with a pre-trained RL model
  - For any $\rho \in [0,1]$, their method is ρ-competitive to the given "expert algorithm"

[LYR23] Pengfei Li, Jianyi Yang, and Shaolei Ren. Learning for edge-weighted online bipartite matching with robustness guarantees. International Conference on Machine Learning (ICML), 2023

Research question

- If we have "perfect information" about G^*, can get n^* matches?
- Also, we know that `Ranking` achieves competitive ratio of $1 - \frac{1}{e}$

Can we get an algorithm that is both 1-consistent and $\left(1 - \frac{1}{e}\right)$-robust?

Our first main result

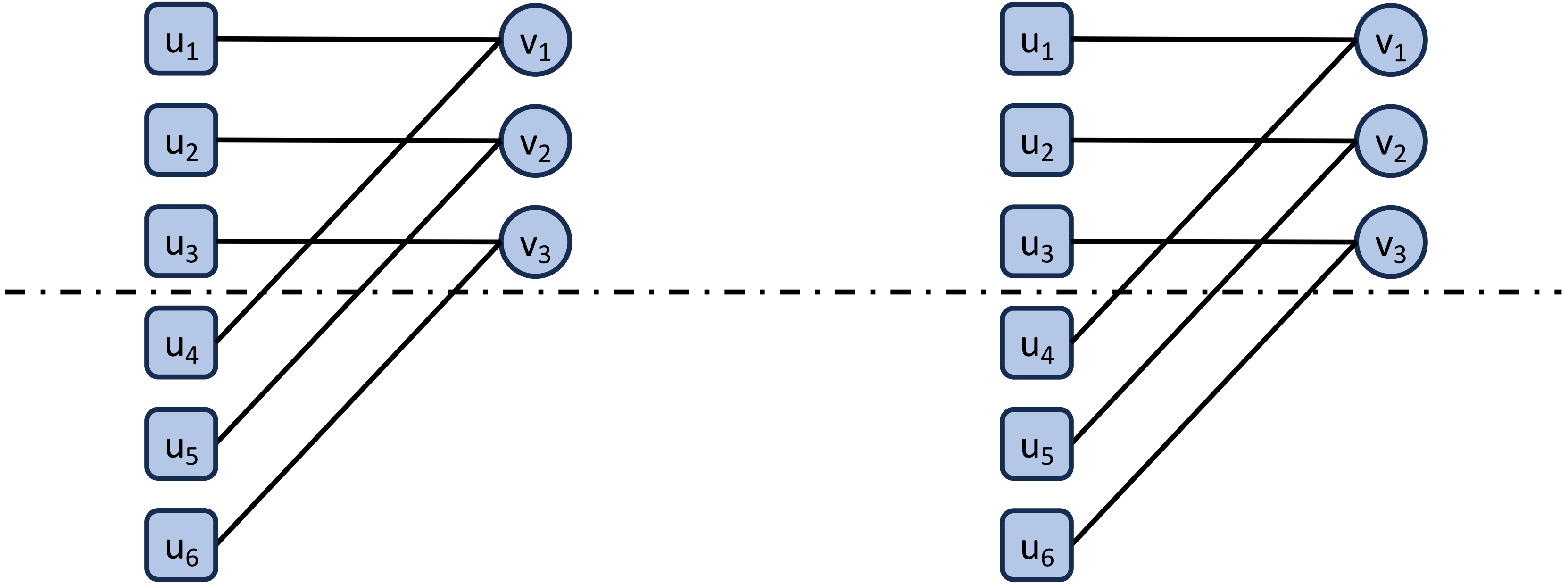
Impossibility result (Informal)

With adversarial vertex arrivals, no algorithm can be both 1-consistent and $> \frac{1}{2}$-robust, regardless of advice.

- Extends to $(1 - a)$-consistent and $\left(\frac{1}{2} + a\right)$-robust, for any $a \in [0, ½]$.
- Proof sketch (for $a = 0$ case):
  - Restrict G^* to be one of two possible graphs (next slide)
  - **Any** advice is equivalent to getting 1 bit of information
  - In first $\frac{n}{2}$ arrivals, no algorithm can distinguish between the two graphs
  - **Any** 1-consistent algorithm must behave as if the advice is perfect initially

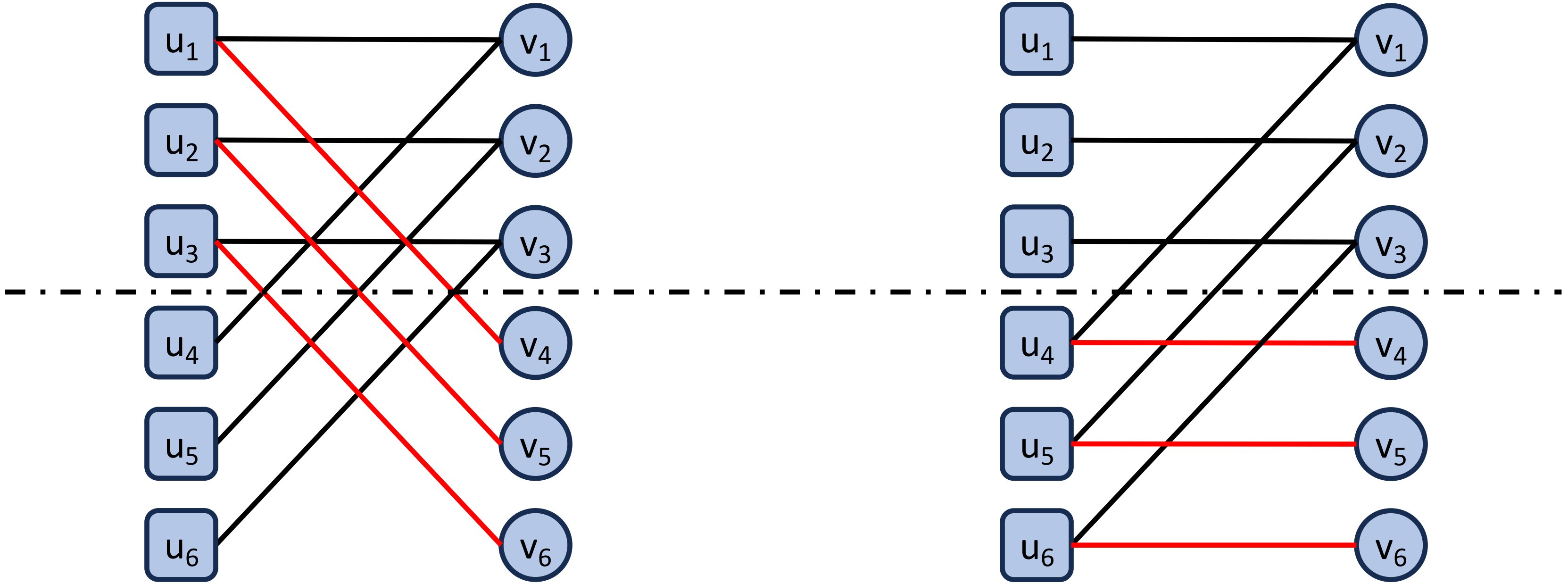
Impossibility result (Informal)

With adversarial vertex arrivals, no algorithm can be both 1-consistent and $> \frac{1}{2}$-robust, regardless of advice.



Impossibility result (Informal)

With adversarial vertex arrivals, no algorithm can be both 1-consistent and $> \frac{1}{2}$-robust, regardless of advice.



Hierarchy of arrival models [M13]

Harder → Easier

Adversarial ≤ Random order ≤ Unknown IID ≤ Known IID

Easier models can achieve higher competitive ratios

[M13] Aranyak Mehta. *Online matching and ad allocation*. Foundations and Trends in Theoretical Computer Science, 2013

Hierarchy of arrival models [M13]

Each online vertex is drawn from some type distribution $\mathcal{D}: 2^U \to \mathbb{R}$ in an IID fashion

Worst case G*

Adversarial ≤ Random order ≤ Unknown IID ≤ Known IID

Worst case arrival sequence

Arrival sequence is a random permutation

$\mathcal{D}: 2^U \to \mathbb{R}$ unknown

$\mathcal{D}: 2^U \to \mathbb{R}$ known

What is known?

Adversarial $\leq$ Random order $\leq$ Unknown IID $\leq$ Known IID

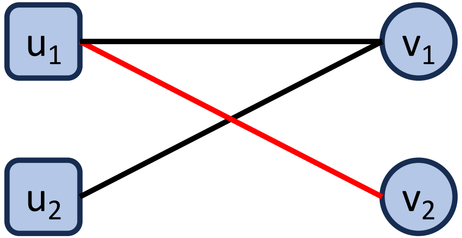
	(Expected) Competitive ratio		
	Adversarial arrival		Random order arrival
Deterministic algorithm	$\frac{1}{2}$	Greedy	$1-\frac{1}{e}$ [GM08]
Deterministic hardness	$\frac{1}{2}$		$\frac{3}{4}$
Randomized algorithm	$1-\frac{1}{e}$ [KVV90]	Ranking	0.696 [MY11]
Randomized hardness	$1-\frac{1}{e}+o(1)$ [KVV90]		0.823 [MGS12]

[GM08] Gagan Goel and Aranyak Mehta. *Online budgeted matching in random input models with applications to Adwords*. Symposium on Discrete Algorithms (SODA), 2008
[MY11] Mohammad Mahdian and Qiqi Yan. *Online Bipartite Matching with Random Arrivals: An Approach Based on Strongly Factor-Revealing LPs*. Symposium on Theory of Computing (STOC), 2011
[MGS12] Vahideh H Manshadi, Shayan Oveis Gharan, and Amin Saberi. *Online stochastic matching: Online actions based on offline statistics*. Mathematics of Operations Research, 2012

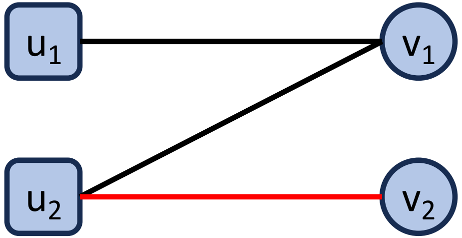
What is known?

	(Expected) Competitive ratio	
	Adversarial arrival	Random order arrival
Deterministic algorithm	$\frac{1}{2}$ Greedy	$1-\frac{1}{e}$ [GM08]
Deterministic hardness	$\frac{1}{2}$	$\frac{3}{4}$
Randomized algorithm	$1-\frac{1}{e}$ [KVV90] Ranking	0.696 [MY11]
Randomized hardness	$1-\frac{1}{e}+o(1)$ [KVV90]	0.823 [MGS12]

- Hardness of $\frac{3}{4}$ for random order



versus



- [KMT11] showed that Ranking cannot beat 0.727 in general
- So, new ideas are needed if you believe the “right bound” is 0.823

[KMT11] Mohammad Mahdian and Qiqi Yan. *Online Bipartite Matching with Random Arrivals: An Approach Based on Strongly Factor-Revealing LPs*. Symposium on Theory of Computing (STOC), 2011

Our second main result

Can we get an algorithm that is both 1-consistent and $\cancel{\left(1-\frac{1}{e}\right)}$ β-robust?

- Let β denote the “best possible competitive ratio”
- Our first result says: This is not possible for adversarial arrivals!
- What about random order arrivals?

Adversarial $\leq$ Random order $\leq$ Unknown IID $\leq$ Known IID

Our second main result

Can we get an algorithm that is both 1-consistent and $\cancel{\left(1-\frac{1}{e}\right)}$ β-robust?

Goal achievable in random order (Informal)

With random order, there is an algorithm achieves competitive ratio interpolating between 1 and $\beta \cdot \big(1 - o(1)\big)$, depending on advice quality.

- Our method is a meta-algorithm that uses any `Baseline` that achieves β
- So, we are simultaneously 1-consistent and $\beta \cdot \big(1 - o(1)\big)$-robust
- For random arrival model, we know that $0.696 \leq \beta \leq 0.823$

e.g. use `Ranking`

Realized type counts as advice

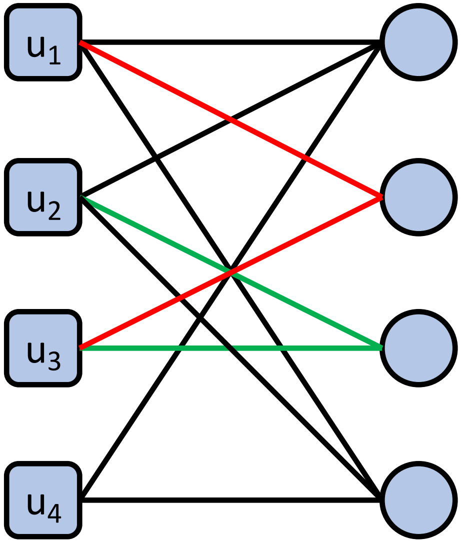
- Classify online vertex in $G^* = (U \cup V, E)$ based on their types
  - Type of v_i is the set of offline vertices in $N(v_i)$ are adjacent to [BKP20]
- Define integer vector $c^* \in \mathbb{N}^{2^n}$ indexed by all possible types 2^U
  - $c^*(t)$ = Number of times the type $t \in 2^U$ occurs in G^*
- Define $T^* \subseteq 2^U$ as the subset of non-zero counts in c^*
  - Note: $|T^*| \leq n \ll 2^{|U|} = 2^n$

[BKP20] Allan Borodin, Christodoulos Karavasilis, and Denis Pankratov. *An experimental study of algorithms for online bipartite matching*. Journal of Experimental Algorithmics (JEA), 2020

Realized type counts as advice

- Classify online vertex in $G^* = (U \cup V, E)$ based on their types
  - Type of v_i is the set of offline vertices in $N(v_i)$ are adjacent to [BKP20]
- Define integer vector $c^* \in \mathbb{N}^{2^n}$ indexed by all possible types 2^U
  - $c^*(t)$ = Number of times the type $t \in 2^U$ occurs in G^*
- Define $T^* \subseteq 2^U$ as the subset of non-zero counts in c^*
  - Note: $|T^*| \leq n \ll 2^{|U|} = 2^n$
- Advice is simply an estimate vector $\hat{c}$ which approximates c^*
  - Let $\widehat{T}$ be non-zero counts in $\hat{c}$. Similarly, we have $\left|\widehat{T}\right| \leq n$
  - Can represent $\hat{c}$ using $O(n)$ labels and numbers

Realized type counts as advice



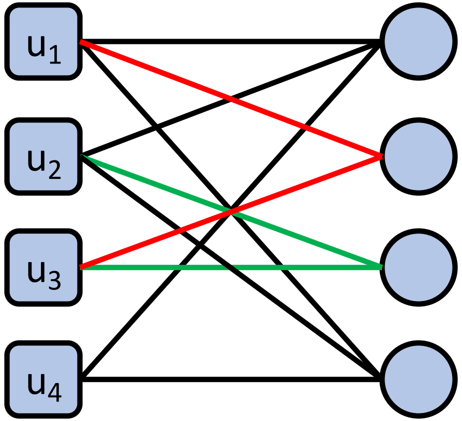
Type	c^*
$\{u_1, u_2, u_4\}$	2
$\{u_1, u_3\}$	1
$\{u_2, u_3\}$	1
$2^U \setminus T^*$	0

T^*

Here, $|T^*| = 3 \ll 2^4 = 16$

The `Mimic` algorithm

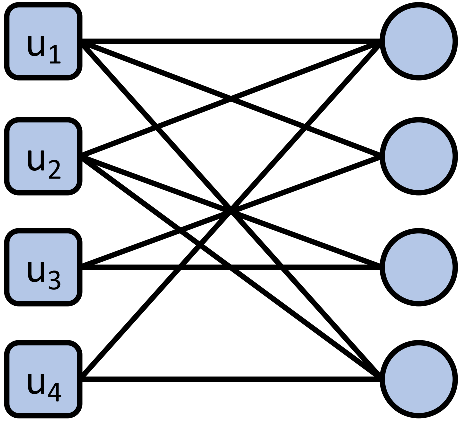
- Algorithm
  - Fix any arbitrary maximum matching $\widehat{M}$ on the graph defined by advice $\hat{c}$
  - Try to mimic edge matches in $\widehat{M}$ while tracking the types of each arrival
  - If unable to mimic, leave arrival unmatched.



Type	c^*
$\{u_1, u_2, u_4\}$	2
$\{u_1, u_3\}$	1
$\{u_2, u_3\}$	1

The `Mimic` algorithm

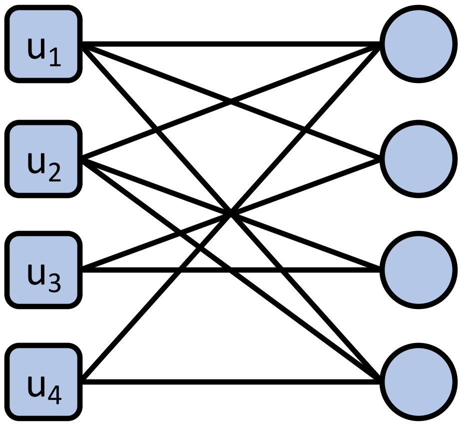
- Algorithm
  - Fix any arbitrary maximum matching $\widehat{M}$ on the graph defined by advice $\hat{c}$
  - Try to mimic edge matches in $\widehat{M}$ while tracking the types of each arrival
  - If unable to mimic, leave arrival unmatched.



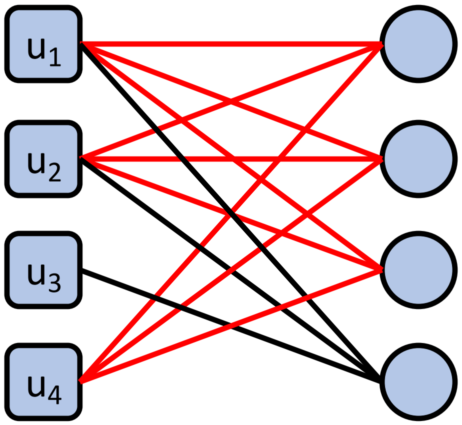
Type	c^*	$\hat{c}$
$\{u_1, u_2, u_4\}$	2	3
$\{u_1, u_3\}$	1	0
$\{u_2, u_3\}$	1	0
$\{u_1, u_2, u_3\}$	0	1

The `Mimic` algorithm

- Algorithm
  - Fix any arbitrary maximum matching $\widehat{M}$ on the graph defined by advice $\hat{c}$
  - Try to mimic edge matches in $\widehat{M}$ while tracking the types of each arrival
  - If unable to mimic, leave arrival unmatched.

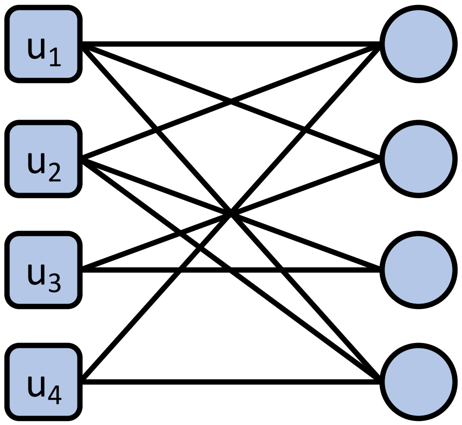


Type	c^*	$\hat{c}$
$\{u_1, u_2, u_4\}$	2	3
$\{u_1, u_3\}$	1	0
$\{u_2, u_3\}$	1	0
$\{u_1, u_2, u_3\}$	0	1

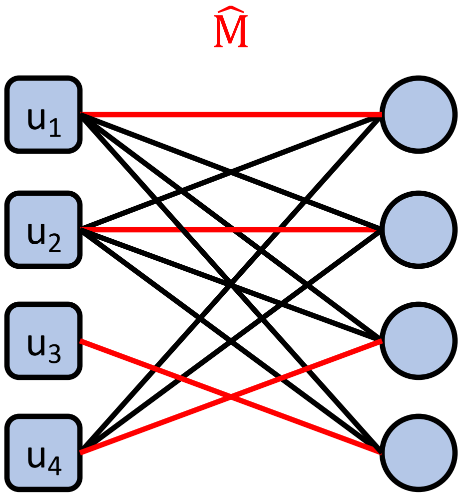


The Mimic algorithm

- Algorithm
  - Fix any arbitrary maximum matching $\widehat{M}$ on the graph defined by advice $\hat{c}$
  - Try to mimic edge matches in $\widehat{M}$ while tracking the types of each arrival
  - If unable to mimic, leave arrival unmatched.

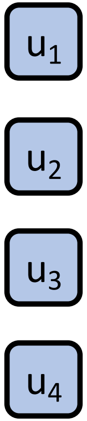


Type	c^*	$\hat{c}$
$\{u_1, u_2, u_4\}$	2	3
$\{u_1, u_3\}$	1	0
$\{u_2, u_3\}$	1	0
$\{u_1, u_2, u_3\}$	0	1

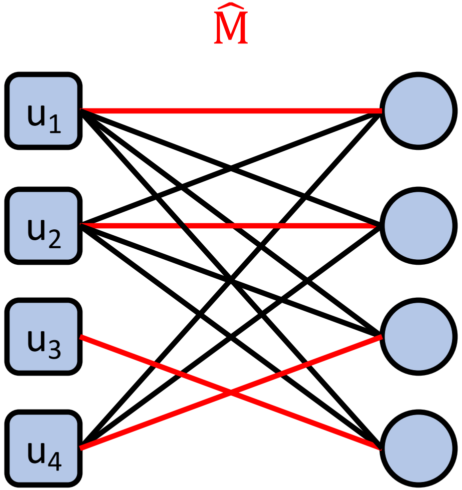


The `Mimic` algorithm

- Algorithm
  - Fix any arbitrary maximum matching $\widehat{M}$ on the graph defined by advice $\hat{c}$
  - Try to mimic edge matches in $\widehat{M}$ while tracking the types of each arrival
  - If unable to mimic, leave arrival unmatched.

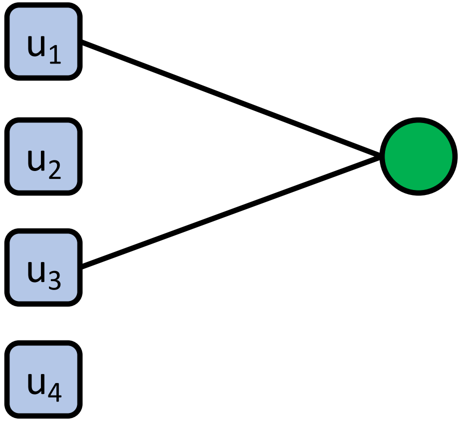


Type	c^*	$\hat{c}$
$\{u_1, u_2, u_4\}$	2	3
$\{u_1, u_3\}$	1	0
$\{u_2, u_3\}$	1	0
$\{u_1, u_2, u_3\}$	0	1

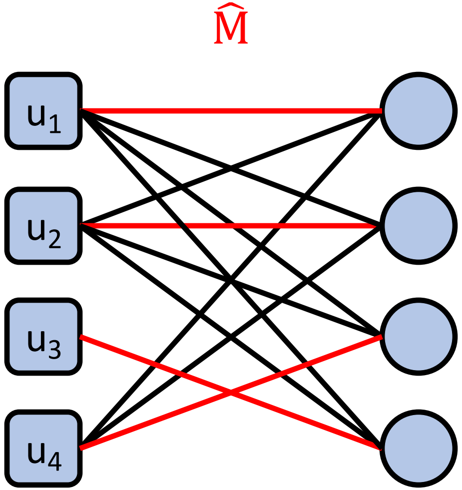


The `Mimic` algorithm

- Algorithm
  - Fix any arbitrary maximum matching $\widehat{M}$ on the graph defined by advice $\hat{c}$
  - Try to mimic edge matches in $\widehat{M}$ while tracking the types of each arrival
  - If unable to mimic, leave arrival unmatched.

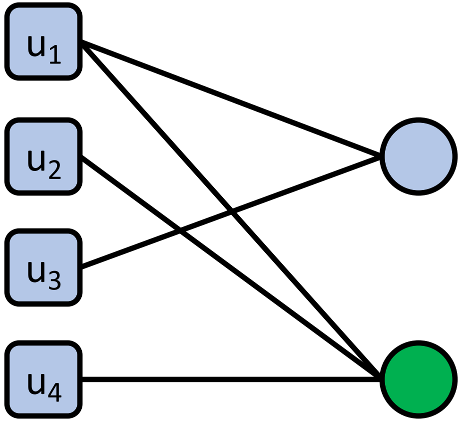


Type	c^*	$\hat{c}$
$\{u_1, u_2, u_4\}$	2	3
$\{u_1, u_3\}$	1	0
$\{u_2, u_3\}$	1	0
$\{u_1, u_2, u_3\}$	0	1

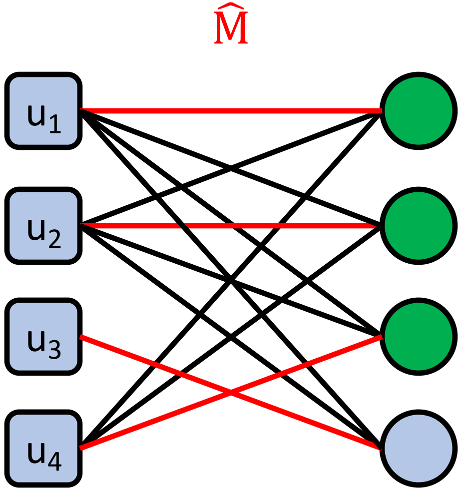


The `Mimic` algorithm

- Algorithm
  - Fix any arbitrary maximum matching $\widehat{M}$ on the graph defined by advice $\hat{c}$
  - Try to mimic edge matches in $\widehat{M}$ while tracking the types of each arrival
  - If unable to mimic, leave arrival unmatched.

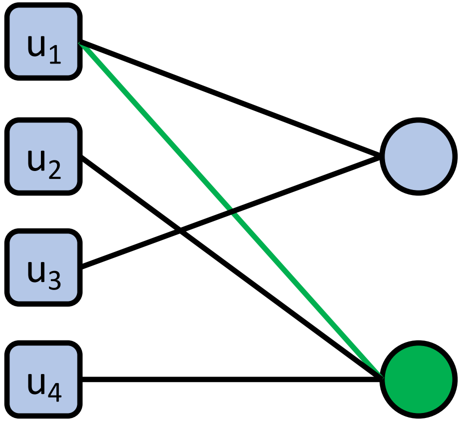


Type	c^*	$\hat{c}$
$\{u_1, u_2, u_4\}$	2	3
$\{u_1, u_3\}$	1	0
$\{u_2, u_3\}$	1	0
$\{u_1, u_2, u_3\}$	0	1



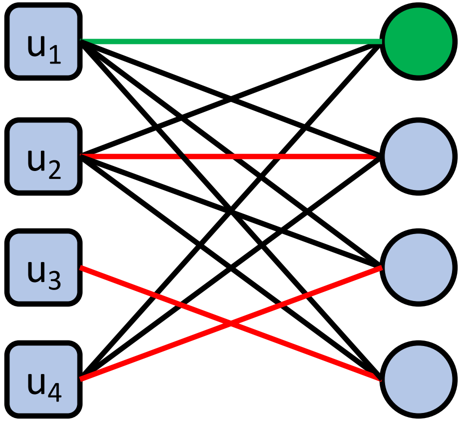
The `Mimic` algorithm

- Algorithm
  - Fix any arbitrary maximum matching $\widehat{M}$ on the graph defined by advice $\hat{c}$
  - Try to mimic edge matches in $\widehat{M}$ while tracking the types of each arrival
  - If unable to mimic, leave arrival unmatched.



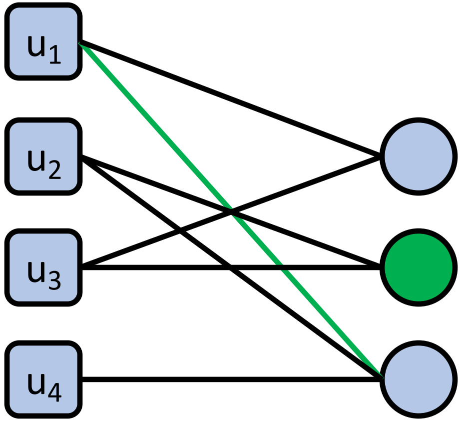
Type	c^*	$\hat{c}$
$\{u_1, u_2, u_4\}$	2	$\cancel{3}$ 2
$\{u_1, u_3\}$	1	0
$\{u_2, u_3\}$	1	0
$\{u_1, u_2, u_3\}$	0	1

$\widehat{M}$



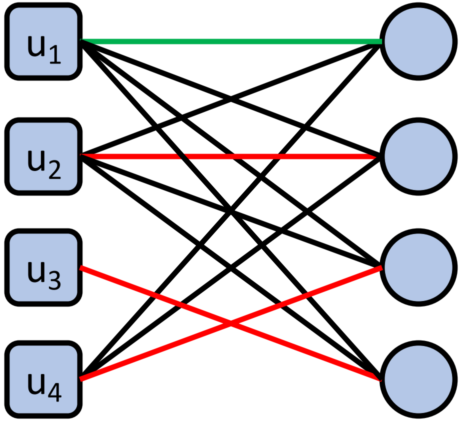
The `Mimic` algorithm

- Algorithm
  - Fix any arbitrary maximum matching $\widehat{M}$ on the graph defined by advice $\hat{c}$
  - Try to mimic edge matches in $\widehat{M}$ while tracking the types of each arrival
  - If unable to mimic, leave arrival unmatched.



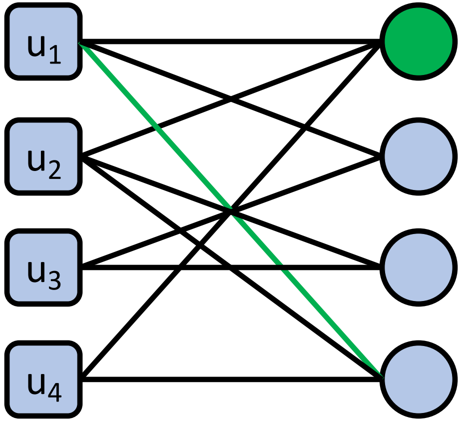
Type	c^*	$\hat{c}$
$\{u_1, u_2, u_4\}$	2	~~3~~ 2
$\{u_1, u_3\}$	1	0
$\{u_2, u_3\}$	1	0
$\{u_1, u_2, u_3\}$	0	1

$\widehat{M}$

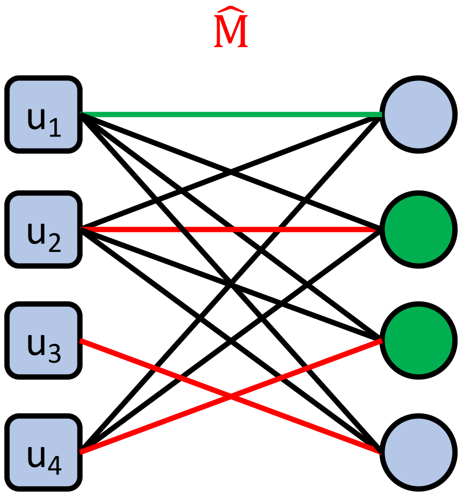


The `Mimic` algorithm

- Algorithm
  - Fix any arbitrary maximum matching $\widehat{M}$ on the graph defined by advice $\hat{c}$
  - Try to mimic edge matches in $\widehat{M}$ while tracking the types of each arrival
  - If unable to mimic, leave arrival unmatched.

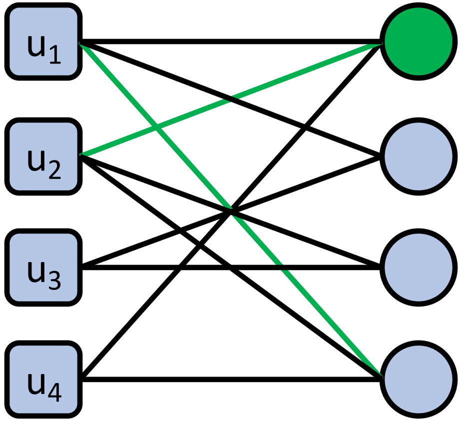


Type	c^*	$\hat{c}$
$\{u_1, u_2, u_4\}$	2	$\cancel{3}$ 2
$\{u_1, u_3\}$	1	0
$\{u_2, u_3\}$	1	0
$\{u_1, u_2, u_3\}$	0	1

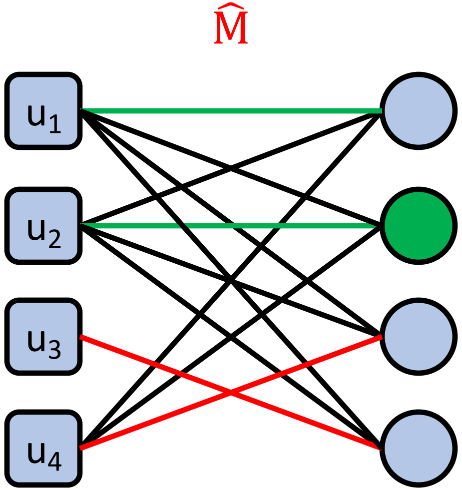


The `Mimic` algorithm

- Algorithm
  - Fix any arbitrary maximum matching $\widehat{M}$ on the graph defined by advice $\hat{c}$
  - Try to mimic edge matches in $\widehat{M}$ while tracking the types of each arrival
  - If unable to mimic, leave arrival unmatched.

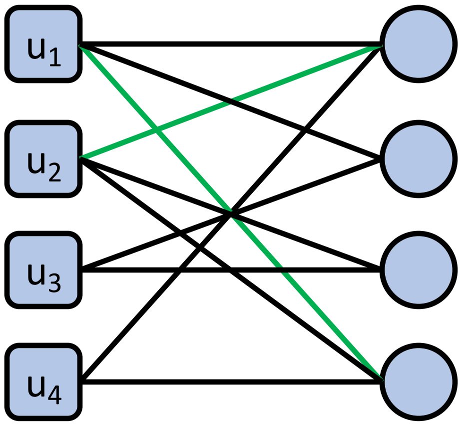


Type	c^*	$\hat{c}$
$\{u_1, u_2, u_4\}$	2	$\cancel{3}$ $\cancel{2}$ 1
$\{u_1, u_3\}$	1	0
$\{u_2, u_3\}$	1	0
$\{u_1, u_2, u_3\}$	0	1



The `Mimic` algorithm

- Algorithm
  - Fix any arbitrary maximum matching $\widehat{M}$ on the graph defined by advice $\hat{c}$
  - Try to mimic edge matches in $\widehat{M}$ while tracking the types of each arrival
  - If unable to mimic, leave arrival unmatched.



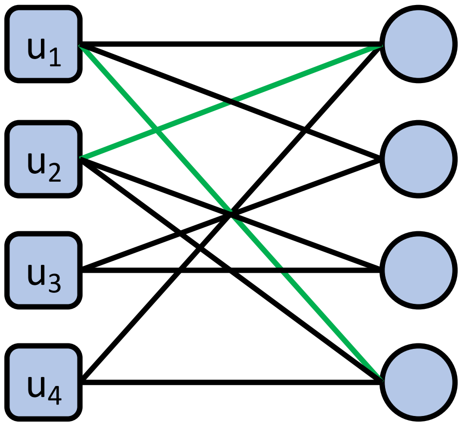
Type	c^*	$\hat{c}$
$\{u_1, u_2, u_4\}$	2	3
$\{u_1, u_3\}$	1	0
$\{u_2, u_3\}$	1	0
$\{u_1, u_2, u_3\}$	0	1

Produced matching size

= 2

The Mimic algorithm

- Algorithm
  - Fix any arbitrary maximum matching $\widehat{M}$ on the graph defined by advice $\hat{c}$
  - Try to mimic edge matches in $\widehat{M}$ while tracking the types of each arrival
  - If unable to mimic, leave arrival unmatched.



Type	c^*	$\hat{c}$
$\{u_1, u_2, u_4\}$	2	3
$\{u_1, u_3\}$	1	0
$\{u_2, u_3\}$	1	0
$\{u_1, u_2, u_3\}$	0	1

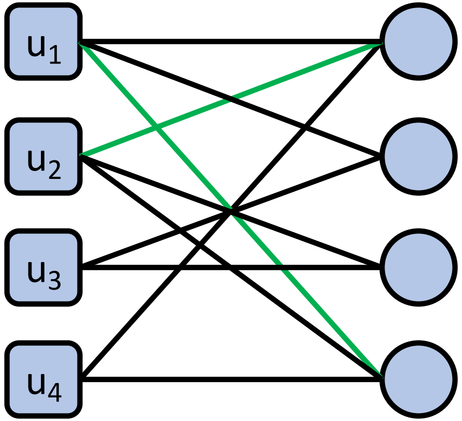
Produced matching size

= 2

$$L_1(c^*, \hat{c}) = |3-2| + |0-1| + |0-1| + |1-0| + 0 \ldots = 4$$

The Mimic algorithm

- Algorithm
  - Fix any arbitrary maximum matching $\widehat{M}$ on the graph defined by advice $\hat{c}$
  - Try to mimic edge matches in $\widehat{M}$ while tracking the types of each arrival
  - If unable to mimic, leave arrival unmatched.



Type	c^*	$\hat{c}$
$\{u_1, u_2, u_4\}$	2	3
$\{u_1, u_3\}$	1	0
$\{u_2, u_3\}$	1	0
$\{u_1, u_2, u_3\}$	0	1

Produced matching size

$$= 2 = \left|\widehat{M}\right| - \frac{L_1(c^*, \hat{c})}{2}$$

Error is "double counted" in L_1

$$L_1(c^*, \hat{c}) = |3-2| + |0-1| + |0-1| + |1-0| + 0 \ldots = 4$$

The Mimic algorithm

- Algorithm
  - Fix any arbitrary maximum matching $\widehat{\mathrm{M}}$ on the graph defined by advice $\hat{\mathrm{c}}$
  - Try to mimic edge matches in $\widehat{\mathrm{M}}$ while tracking the types of each arrival
  - If unable to mimic, leave arrival unmatched.
- Analysis
  - $0 \leq \mathrm{L}_1(\mathrm{c}^*, \hat{\mathrm{c}}) \leq 2\mathrm{n}$ measures how close $\hat{\mathrm{c}}$ is to c^*
  - By blindly following advice, Mimic gets a matching of size $|\widehat{\mathrm{M}}| - \frac{\mathrm{L}_1(\mathrm{c}^*, \hat{\mathrm{c}})}{2}$
  - Mimic beats an advice-free Baseline whenever $|\widehat{\mathrm{M}}| - \frac{\mathrm{L}_1(\mathrm{c}^*, \hat{\mathrm{c}})}{2} > \beta \cdot \mathrm{n}$

The Mimic algorithm

- Algorithm
  - Fix any arbitrary maximum matching $\widehat{M}$ on the graph defined by advice $\hat{c}$
  - Try to mimic edge matches in $\widehat{M}$ while tracking the types of each arrival
  - If unable to mimic, leave arrival unmatched.
- Analysis
  - $0 \leq L_1(c^*, \hat{c}) \leq 2n$ measures how close $\hat{c}$ is to c^*
  - By blindly following advice, Mimic gets a matching of size $|\widehat{M}| - \frac{L_1(c^*,\hat{c})}{2}$
  - Mimic beats an advice-free Baseline whenever $|\widehat{M}| - \frac{L_1(c^*,\hat{c})}{2} > \beta \cdot n$
  - Mimic beats an advice-free Baseline whenever $\frac{L_1(c^*,\hat{c})}{n} < 2(1 - \beta)$

For this talk, let's treat $|\widehat{M}| = n$

How to test advice quality?

Insight: Use sublinear property testing to estimate $L_1(c^*, \hat{c})$!

- Define $p = \frac{c^*}{n}$ and $q = \frac{\hat{c}}{n}$ as distributions over the 2^U types

How to test advice quality?

Insight: Use sublinear property testing to estimate $L_1(c^*, \hat{c})$!

- Define $p = \frac{c^*}{n}$ and $q = \frac{\hat{c}}{n}$ as distributions over the 2^U types
- [VV11, JHW18]: Can estimate $L_1(p, q)$ "well" using $o(n)$ IID samples
  - To be precise, if p and q have domain size $r \leq n$, then $\Theta\left(\frac{r}{\varepsilon^2 \log r}\right)$ IID samples sufficient and necessary to estimate $\hat{L}_1$ such that $\left|\hat{L}_1 - L_1(p, q)\right| \leq \varepsilon$
  - c^* and $\hat{c}$ can be defined over $\left|\hat{T}\right| + 1$ elements with a "not in $\hat{T}$" bucket

[VV11] Gregory Valiant and Paul Valiant. *The power of linear estimators*. Foundations of Computer Science (FOCS), 2011.
[JHW18] Jiantao Jiao, Yanjun Han, and Tsachy Weissman. *Minimax estimation of the L_1 distance*. IEEE Transactions on Information Theory, 2018.

Some minor adjustments to our problem setting

- Adjustment 1
  - Random vertex arrivals are “sampling without replacement”
  - We can simulate IID samples by keeping track of what has arrived and then “reusing” arrivals with some probability proportional to number of arrivals
- Adjustment 2
  - L_1 estimator is in expectation, but can be made “with high probability”

The `TestAndMatch` algorithm

- Algorithm
  - Fix any arbitrary maximum matching $\widehat{\mathrm{M}}$ on the graph defined by advice $\hat{\mathrm{c}}$
  - If $|\widehat{\mathrm{M}}| \leq \beta \cdot \mathrm{n}$, run the best advice-free `Baseline` on all arrivals
  - Otherwise, run `Mimic` while testing quality of $\hat{\mathrm{c}}$ by estimating $\mathrm{L}_1(\mathrm{c}^*, \hat{\mathrm{c}})$
  - If test declares $\mathrm{L}_1(\mathrm{c}^*, \hat{\mathrm{c}})$ is “large”, use `Baseline` for remaining arrivals
  - Otherwise, continue using `Mimic` for remaining arrivals

The `TestAndMatch` algorithm

- Algorithm
  - Fix any arbitrary maximum matching $\widehat{M}$ on the graph defined by advice $\hat{c}$
  - If $|\widehat{M}| \leq \beta \cdot n$, run the best advice-free `Baseline` on all arrivals
  - Otherwise, run `Mimic` while testing quality of $\hat{c}$ by estimating $L_1(c^*, \hat{c})$
  - If test declares $L_1(c^*, \hat{c})$ is "large", use `Baseline` for remaining arrivals
  - Otherwise, continue using `Mimic` for remaining arrivals
- Analysis
  - If $\hat{L}_1 \lesssim 2(1-\beta)$, then `TestAndMatch` attains ratio of at least $1 - \frac{L_1(c^*,\hat{c})}{2n}$
  - Otherwise, `TestAndMatch` attains ratio of at least $\beta \cdot (1 - o(1))$

Our second main result

Can we get an algorithm that is both 1-consistent and $\cancel{\left(1-\frac{1}{e}\right)}$ β-robust?

Goal achievable in random order (Informal)

With random order, there is an algorithm achieves competitive ratio interpolating between 1 and $\beta \cdot \left(1 - o(1)\right)$, depending on advice quality.

- Our method is a meta-algorithm that uses any `Baseline` that achieves β
- So, we are simultaneously 1-consistent and $\beta \cdot \left(1 - o(1)\right)$-robust
- For random arrival model, we know that $0.696 \leq \beta \leq 0.823$

Our second main result

Can we get an algorithm that is both 1-consistent and $\cancel{\left(1-\frac{1}{e}\right)}$ β-robust?

Goal achievable in random order (Informal)

Let $\hat{\mathrm{L}}_1$ be estimate of $\mathrm{L}_1(\mathrm{c}^*, \hat{\mathrm{c}})$ from $\mathrm{o}(\mathrm{n})$ vertex arrivals.
TestAndMatch achieves a competitive ratio of at least

- At least $1 - \frac{\mathrm{L}_1(\mathrm{c}^*, \hat{\mathrm{c}})}{2\mathrm{n}} \geq \beta$, when $\hat{\mathrm{L}}_1$ "small"
- At least $\beta \cdot \left(1 - \mathrm{o}(1)\right)$, when $\hat{\mathrm{L}}_1$ "large"

i.e., TestAndMatch is 1-consistent and $\beta \cdot \left(1 - \mathrm{o}(1)\right)$-robust

Conclusions and future directions

- Our paper also discussed some practical considerations while using the given advice $\hat{c}$
- Can our ideas such as using property testing extend to other versions of online bipartite matching and other online problems with random arrivals?
  - We suspect it extends with suitably chosen advice and quality metrics, e.g. Earthmover distance?
- Is there a smarter way using advice other than `Mimic`, leaving some arrivals unmatched?
  - [FMMM09] constructed two matchings to "load balance" in the known IID setting
  - In semi-online model, [KPSSV19] mimic matching on known arrivals and `Ranking` on adversarial arrivals
- Message to the learning-augmented community: Beyond consistency and robustness?
  - `TestAndMatch`'s guarantees is based on L_1 over the type histograms
  - This is sensitive to certain types of noise, e.g. $\hat{c}$ obtained after Erdős–Rényi edits to the offline graph G^*
  - We expect large L_1 in practice, but notions of advice practicality are not formally considered under the standard framework of consistency and robustness

Thank you for your kind attention!

[FMMM09] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and Shan Muthukrishnan. *Online stochastic matching: Beating 1-1/e*. Foundations of Computer Science (FOCS), 2009.
[KPSSV19] Ravi Kumar, Manish Purohit, Aaron Schild, Zoya Svitkina, and Erik Vee. *Semi-Online Bipartite Matching*. Innovations in Theoretical Computer Science (ITCS), 2019.