Single-Sample Prophet Inequalities via Greedy-Ordered Selection

Matthew Faw (UT Austin)

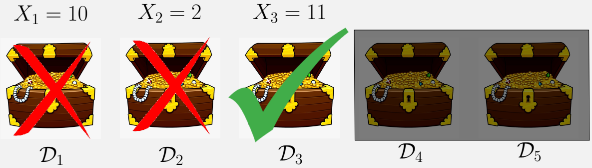
January 10, 2022

Joint work with C.Caramanis, P.Dütting, F.Fusco, P.Lazos, S.Leonardi, O.Papadigenopoulos, E.Pountourakis, and R.Reiffenhäuser

Gambling Against a Prophet

Single-Sample Prophet Inequalities (SSPIs) are a simple variation on the classic *Prophet Inequality* problem:

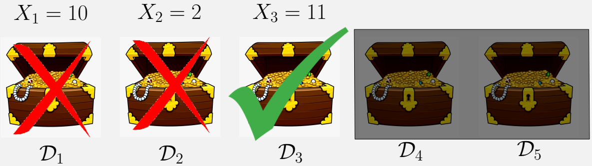
- N rewards $X_i \sim \mathcal{D}_i$ arrive one at a time
- Gambler must *irrevocably* decide whether to:
  a) Collect X_i and end the game, or
  b) Forfeit X_i and continue the game



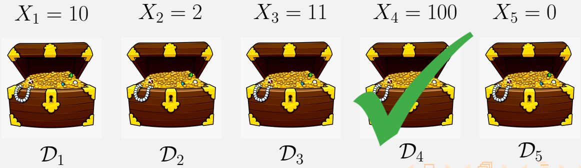
Gambling Against a Prophet

Single-Sample Prophet Inequalities (SSPIs) are a simple variation on the classic *Prophet Inequality* problem:

- N rewards $X_i \sim \mathcal{D}_i$ arrive one at a time
- Gambler must *irrevocably* decide whether to:
  a) Collect X_i and end the game, or
  b) Forfeit X_i and continue the game



- Prophet collects *largest* reward $\max_{i \in [N]} X_i$.



Gambling Against a Prophet

Single-Sample Prophet Inequalities (SSPIs) are a simple variation on the classic *Prophet Inequality* problem:

- N rewards $X_i \sim \mathcal{D}_i$ arrive one at a time
- Gambler must *irrevocably* decide whether to:
  a) Collect X_i and end the game, or
  b) Forfeit X_i and continue the game
- Prophet collects *largest* reward $\max_{i \in [N]} X_i$.
- **Goal**: Maximize expected reward collected by Gambler, relative to that of an *all-knowing* Prophet.

Gambling Against a Prophet

Single-Sample Prophet Inequalities (SSPIs) are a simple variation on the classic *Prophet Inequality* problem:

- N rewards $X_i \sim \mathcal{D}_i$ arrive one at a time
- Gambler must *irrevocably* decide whether to:
  a) Collect X_i and end the game, or
  b) Forfeit X_i and continue the game
- Prophet collects *largest* reward $\max_{i\in[N]} X_i$.
- **Goal**: Maximize expected reward collected by Gambler, relative to that of an *all-knowing* Prophet.
  - i.e., design an "α-competitive" Gambler:

$$\inf_{\mathcal{D}=\mathcal{D}_1\times\ldots\times\mathcal{D}_N} \frac{\mathbb{E}_{\mathcal{D}}\left[\mathit{Gambler}\right]}{\mathbb{E}_{\mathcal{D}}\left[\mathit{Prophet}\right]} \geq \frac{1}{\alpha}.$$

for *smallest possible* $\alpha \geq 1$

Gambling Against a Prophet

Single-Sample Prophet Inequalities (SSPIs) are a simple variation on the classic *Prophet Inequality* problem:

Notable Results:

- $\exists$ a 2-competitive *threshold-based* Gambler policy
- **No** policy can be < 2-competitive
- *But* need to know all *distributions* to compute these thresholds...

Gambling Against a Prophet

Single-Sample Prophet Inequalities (SSPIs) are a simple variation on the classic *Prophet Inequality* problem:

Notable Results:

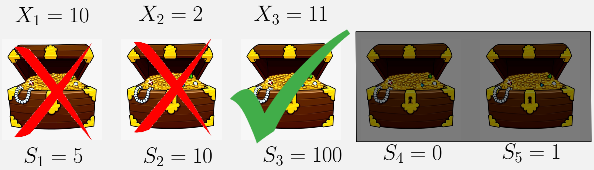
- $\exists$ a 2-competitive *threshold-based* Gambler policy
- **No** policy can be < 2-competitive
- *But* need to know all *distributions* to compute these thresholds...

Question

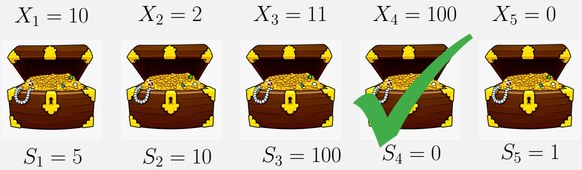
What (if anything) can a Gambler do if she has only a **single sample** *from each* $\mathcal{D}_i$*?*

Gambling Against a Prophet with a *Single* Sample

- N samples $S_i \sim \mathcal{D}_i$ given, rewards $X_i \sim \mathcal{D}_i$ arrive one at a time
- Gambler must *irrevocably* decide whether to:
  a) Collect X_i and end the game, or
  b) Forfeit X_i and continue the game



- Prophet collects *largest* reward $\max_{i \in [N]} X_i$.



Gambling Against a Prophet with a *Single* Sample

- N samples $S_i \sim \mathcal{D}_i$ given, rewards $X_i \sim \mathcal{D}_i$ arrive one at a time
- Gambler must *irrevocably* decide whether to:
  a) Collect X_i and end the game, or
  b) Forfeit X_i and continue the game
- Prophet collects *largest* reward $\max_{i \in [N]} X_i$.
- Perhaps surprisingly, Rubenstein, Wang, and Weinberg (ITCS'20) proved that $\exists$ a 2-competitive (hence *optimal*) **single-sample** policy:
  - Accept the first reward $\geq \tau = \max_i S_i$

Beyond Single-Choice Prophet Inequalities

- Rewards can be collected subject to **combinatorial constraints**:
  - Matroid (e.g., choose k, spanning trees, ...)
  - Matchings
  - Combinatorial Auctions
- Optimal policies are known for some of these settings (e.g., matroids), but require *distributional* knowledge...

Beyond Single-Choice Prophet Inequalities

- Rewards can be collected subject to **combinatorial constraints**:
  - Matroid (e.g., choose k, spanning trees, ...)
  - Matchings
  - Combinatorial Auctions
- Optimal policies are known for some of these settings (e.g., matroids), but require *distributional* knowledge...
- Nearly all **single-sample** prophet inequalities (SSPIs) come via a reduction *to* **order-oblivious secretaries** (OOSs)

Beyond Single-Choice Prophet Inequalities

- Rewards can be collected subject to **combinatorial constraints**:
  - Matroid (e.g., choose k, spanning trees, ...)
  - Matchings
  - Combinatorial Auctions
- Optimal policies are known for some of these settings (e.g., matroids), but require *distributional* knowledge...
- Nearly all **single-sample** prophet inequalities (SSPIs) come via a reduction *to* **order-oblivious secretaries** (OOSs)
- However, this reduction is *necessarily* lossy:
  - Some rewards and samples are *never* used/observed by the policy
  - Leads to *inherently* suboptimal competitive guarantees

Beyond Single-Choice Prophet Inequalities

- Rewards can be collected subject to **combinatorial constraints**:
  - Matroid (e.g., choose k, spanning trees, ...)
  - Matchings
  - Combinatorial Auctions
- Optimal policies are known for some of these settings (e.g., matroids), but require *distributional* knowledge...
- Nearly all **single-sample** prophet inequalities (SSPIs) come via a reduction *to* **order-oblivious secretaries** (OOSs)
- However, this reduction is *necessarily* lossy:
  - Some rewards and samples are *never* used/observed by the policy
  - Leads to *inherently* suboptimal competitive guarantees

Can we do better??

Our Contributions

1. Analyze SSPIs *beyond* single-choice **directly**, *without* reducing to OOS, via an idea we term **greedy-ordered selection**
2. Identify a common property of matroids exploited in many OOS algorithms — a *partition property* — which can be used (together with the *optimal* single-choice SSPI) to obtain improved competitive guarantees
3. Discuss some interesting new connections between SSPIs and OOSs

Our Contributions

Our focus today:

1. Analyze SSPIs *beyond* single-choice **directly**, *without* reducing to OOS, via an idea we term **greedy-ordered selection**
   - Specifically for the case of *matching with edge arrivals*
2. Identify a common property of matroids exploited in many OOS algorithms — a *partition property* — which can be used (together with the *optimal* single-choice SSPI) to obtain improved competitive guarantees
3. Discuss some interesting new connections between SSPIs and OOSs

Some Notable Results

	Combinatorial set	Previous best	Our results
Greedy-ordered selection	General matching (edge arrivals)	512	16
	Budget-additive combinatorial auction	N/A	24
	Bipartite matching (edge arrivals)	256	16
		6.75 (degree-d)	16
		$\mathcal{O}(d^2)$-samples	1 sample
	Bipartite matching (vertex arrivals)	13.5	8
	Transversal matroid	16	8
Partition property	Graphic matroid	8	4
	Co-graphic matroid	12	6
	Low density matroid	$4\gamma(M)$	$2\gamma(M)$
	Column k-sparse linear matroid	$4k$	$2k$

Table: Summary of main results

Main Idea: Greedy-Ordered Selection

Our main results (for matchings and combinatorial auctions) are obtained through a framework we call **greedy-ordered selection**. The general technique is:

1. Design a threshold-based Gambler policy
2. Couple this policy to an *equivalent* "offline" algorithm which traverses samples *and* rewards *together* in decreasing order of weight
3. Guarantee that "important" elements are "typically" selected by this offline algorithm (which is *easier to analyze*)

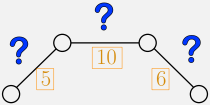
Greedily Gambling Against a Prophet with a *Single* Sample

The greedy *online* policy for matching with edge arrivals:

- Samples $S_e \sim \mathcal{D}_e$ for each edge e given, rewards $X_e \sim \mathcal{D}_e$ arrive one at a time
  - Offline, compute the greedy (maximal) matching M_S on the *samples*
  - Set a threshold τ_e to be the weight of the **heaviest edge in M_S adjacent to** e.
  - Online, accept each arriving edge e if it is feasible and $X_e \geq \tau_e$.

Matching under edge arrivals with a sample

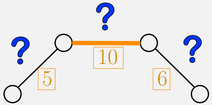
Offline (samples shown below in orange)



Matching under edge arrivals with a sample

Offline (samples shown below in orange, threshold $\tau_e = 10$ for every edge)

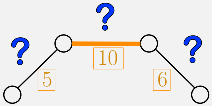
- Compute greedy matching on *samples* M_S



Matching under edge arrivals with a sample

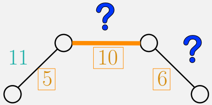
Offline (samples shown below in orange, threshold $\tau_e = 10$ for every edge)

- Compute greedy matching on *samples* M_S



Online (rewards shown above in blue)

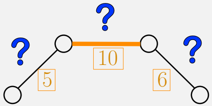
- Rewards X_e arrive one at a time



Matching under edge arrivals with a sample

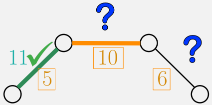
Offline (samples shown below in orange, threshold $\tau_e = 10$ for every edge)

- Compute greedy matching on *samples* M_S



Online (rewards shown above in blue)

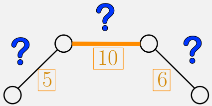
- Rewards X_e arrive one at a time
- Accept an arriving edge e iff it is:
  1. Feasible
  2. Above τ_e, the heaviest edge in M_S adjacent to e



Matching under edge arrivals with a sample

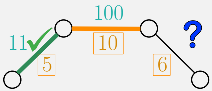
Offline (samples shown below in orange, threshold $\tau_e = 10$ for every edge)

- Compute greedy matching on *samples* M_S



Online (rewards shown above in blue)

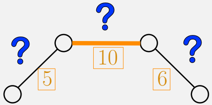
- Rewards X_e arrive one at a time
- Accept an arriving edge e iff it is:
  1. Feasible
  2. Above τ_e, the heaviest edge in M_S adjacent to e



Matching under edge arrivals with a sample

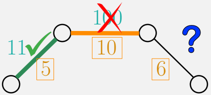
Offline (samples shown below in orange, threshold $\tau_e = 10$ for every edge)

- Compute greedy matching on *samples* M_S



Online (rewards shown above in blue)

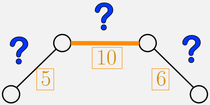
- Rewards X_e arrive one at a time
- Accept an arriving edge e iff it is:
  1. Feasible
  2. Above τ_e, the heaviest edge in M_S adjacent to e



Matching under edge arrivals with a sample

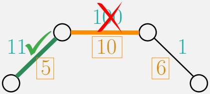
Offline (samples shown below in orange, threshold $\tau_e = 10$ for every edge)

- Compute greedy matching on *samples* M_S



Online (rewards shown above in blue)

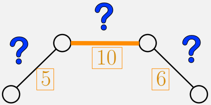
- Rewards X_e arrive one at a time
- Accept an arriving edge e iff it is:
  1. Feasible
  2. Above τ_e, the heaviest edge in M_S adjacent to e



Matching under edge arrivals with a sample

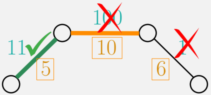
Offline (samples shown below in orange, threshold $\tau_e = 10$ for every edge)

- Compute greedy matching on *samples* M_S



Online (rewards shown above in blue)

- Rewards X_e arrive one at a time
- Accept an arriving edge e iff it is:
  1. Feasible
  2. Above τ_e, the heaviest edge in M_S adjacent to e



An equivalent offline simulation

The equivalent *offline* policy for matching with edge arrivals:

- Deferred decisions (offline):
  - (Conceptually) generate 2 "anonymous" values $V_{1,e}, V_{2,e} \sim \mathcal{D}_e$ for each edge e (relabel s.t. $V_{1,e} > V_{2,e}$)
  - Greedily traverse these 2n values:
    1. When $V_{1,e}$ is encountered, flip a *fair coin* to determine "status" (reward/sample)
    2. When $V_{2,e}$ is encountered, set its status to *opposite* of $V_{1,e}$

An equivalent offline simulation

The equivalent *offline* policy for matching with edge arrivals:

- Deferred decisions (offline):
  - (Conceptually) generate 2 "anonymous" values $V_{1,e}, V_{2,e} \sim \mathcal{D}_e$ for each edge e (relabel s.t. $V_{1,e} > V_{2,e}$)
  - Greedily traverse these 2n values:
    1. When $V_{1,e}$ is encountered, flip a *fair coin* to determine "status" (reward/sample)
    2. When $V_{2,e}$ is encountered, set its status to *opposite* of $V_{1,e}$
- Compute the greedy sample solution, M_S, and the thresholds, τ_e, *exactly* as before

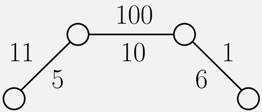
An equivalent offline simulation

The equivalent *offline* policy for matching with edge arrivals:

- Deferred decisions (offline):
  - (Conceptually) generate 2 "anonymous" values $V_{1,e}, V_{2,e} \sim \mathcal{D}_e$ for each edge e (relabel s.t. $V_{1,e} > V_{2,e}$)
  - Greedily traverse these 2n values:
    1. When $V_{1,e}$ is encountered, flip a *fair coin* to determine "status" (reward/sample)
    2. When $V_{2,e}$ is encountered, set its status to *opposite* of $V_{1,e}$
- Compute the greedy sample solution, M_S, and the thresholds, τ_e, *exactly* as before
- Accept elements online *exactly* as before

Matching under edge arrivals with a sample

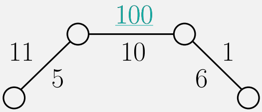
Offline (2n "anonymous" values below and above)



Matching under edge arrivals with a sample

Offline ($2n$ "anonymous" values below and above)

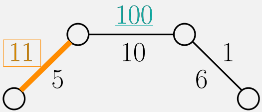
- The largest value for *each edge* e, $V_{1,e}$, assigned as reward or sample w.p. $1/2$



Matching under edge arrivals with a sample

Offline ($2n$ "anonymous" values below and above)

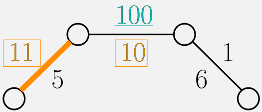
- The largest value for *each edge* e, $V_{1,e}$, assigned as reward or sample w.p. $1/2$



Matching under edge arrivals with a sample

Offline ($2n$ "anonymous" values below and above)

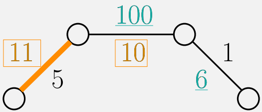
- The largest value for *each edge* e, $V_{1,e}$, assigned as reward or sample w.p. $1/2$
- The smaller value for *each edge* e, $V_{2,e}$ is assigned the *opposite*, deterministically



Matching under edge arrivals with a sample

Offline (2n "anonymous" values below and above)

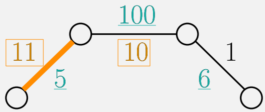
- The largest value for *each edge* e, $V_{1,e}$, assigned as reward or sample w.p. $1/2$
- The smaller value for *each edge* e, $V_{2,e}$ is assigned the *opposite*, deterministically



Matching under edge arrivals with a sample

Offline ($2n$ "anonymous" values below and above)

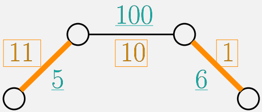
- The largest value for *each edge* e, $V_{1,e}$, assigned as reward or sample w.p. $1/2$
- The smaller value for *each edge* e, $V_{2,e}$ is assigned the *opposite*, deterministically



Matching under edge arrivals with a sample

Offline ($2n$ "anonymous" values below and above)

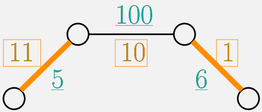
- The largest value for *each edge e*, $V_{1,e}$, assigned as reward or sample w.p. $1/2$
- The smaller value for *each edge e*, $V_{2,e}$ is assigned the *opposite*, deterministically



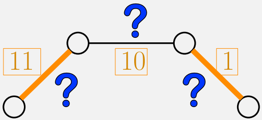
Matching under edge arrivals with a sample

Offline (2n "anonymous" values below and above)

- The largest value for *each edge* e, $V_{1,e}$, assigned as reward or sample w.p. $1/2$
- The smaller value for *each edge* e, $V_{2,e}$ is assigned the *opposite*, deterministically



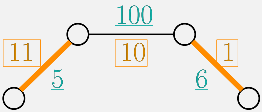
Online (rewards revealed sequentially to Gambler)



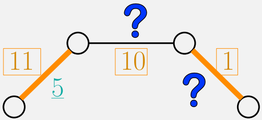
Matching under edge arrivals with a sample

Offline ($2n$ "anonymous" values below and above)

- The largest value for *each edge* e, $V_{1,e}$, assigned as reward or sample w.p. $1/2$
- The smaller value for *each edge* e, $V_{2,e}$ is assigned the *opposite*, deterministically



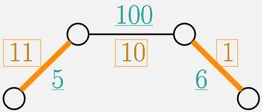
Online (rewards revealed sequentially to Gambler)



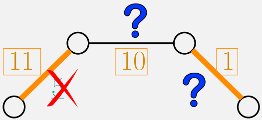
Matching under edge arrivals with a sample

Offline ($2n$ "anonymous" values below and above)

- The largest value for *each edge e*, $V_{1,e}$, assigned as reward or sample w.p. $1/2$
- The smaller value for *each edge e*, $V_{2,e}$ is assigned the *opposite*, deterministically



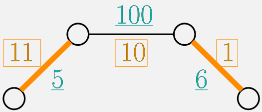
Online (rewards revealed sequentially to Gambler)



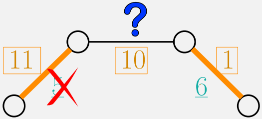
Matching under edge arrivals with a sample

Offline ($2n$ "anonymous" values below and above)

- The largest value for *each edge* e, $V_{1,e}$, assigned as reward or sample w.p. $1/2$
- The smaller value for *each edge* e, $V_{2,e}$ is assigned the *opposite*, deterministically



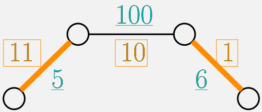
Online (rewards revealed sequentially to Gambler)



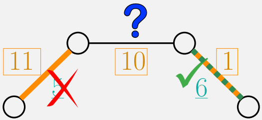
Matching under edge arrivals with a sample

Offline ($2n$ "anonymous" values below and above)

- The largest value for *each edge* e, $V_{1,e}$, assigned as reward or sample w.p. $1/2$
- The smaller value for *each edge* e, $V_{2,e}$ is assigned the *opposite*, deterministically



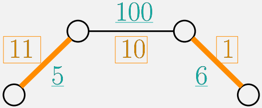
Online (rewards revealed sequentially to Gambler)



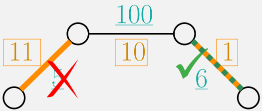
Matching under edge arrivals with a sample

Offline ($2n$ "anonymous" values below and above)

- The largest value for *each edge e*, $V_{1,e}$, assigned as reward or sample w.p. $1/2$
- The smaller value for *each edge e*, $V_{2,e}$ is assigned the *opposite*, deterministically



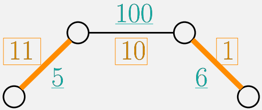
Online (rewards revealed sequentially to Gambler)



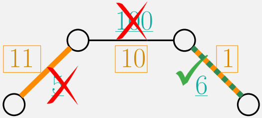
Matching under edge arrivals with a sample

Offline ($2n$ "anonymous" values below and above)

- The largest value for *each edge e*, $V_{1,e}$, assigned as reward or sample w.p. $1/2$
- The smaller value for *each edge e*, $V_{2,e}$ is assigned the *opposite*, deterministically



Online (rewards revealed sequentially to Gambler)



Note

- This *equivalent algorithm* viewpoint has been exploited in the secretary algorithm literature (e.g., Korula and Pal '09, Ma, Tang, and Wang '11)
- Key difficulty in our setting:
  - The status of the $V_{2,i}$'s (i.e., whether they are a sample or reward) are *correlated* with status of the corresponding $V_{1,i}$

Safe Elements

A key idea of our proofs is constructing a set of “heavy” elements which are (effectively) guaranteed to be collected, *even* in *adversarial* order

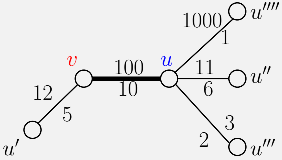
Safe Edges (Matching with Edge Arrivals)

We can an edge $e = \{v, u\}$ *"safe" for a vertex* v if:

1. $V_{1,e}$ is a **reward** that would be in the greedy solution *w.r.t. samples*, *if it were a sample*
   - "Could be in the greedy solution"
2. No edge neighboring v can block e from being accepted
   - "No conflicts with v"
3. No edge of *smaller weight* than e neighboring u can block e from being accepted
   - "No *small* conflicts with u"

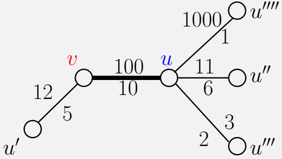
Manipulating safe edges via greedy-ordered selection

Let's see how, through **greedy-ordered selection**, to ensure that a "heavy" edge is "safe"



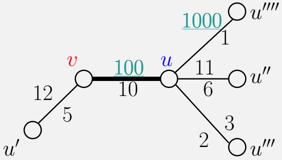
Manipulating safe edges via greedy-ordered selection

Rewards in blue, samples in orange. Initially, all values are **anonymous**.
Goal: make $e = \{v, u\}$ *safe for* v.



Manipulating safe edges via greedy-ordered selection

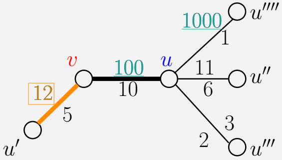
Rewards in blue, samples in orange. Initially, all values are **anonymous**.
Goal: make $e = \{v, u\}$ *safe for* v.



- Start with any run of the greedy offline algorithm (which determines sample/reward status of the $2n$ values) such that:
  1. $e = \{v, u\}$ could be added to the **greedy solution w.r.t. samples**, *if* $V_{1,e} = 100$ *were a sample*
  2. $V_{1,e} = 100$ is a **reward**

Manipulating safe edges via greedy-ordered selection

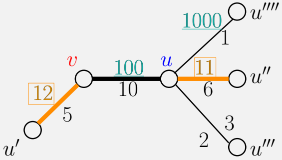
Rewards in blue, samples in orange. Initially, all values are **anonymous**.
Goal: make $e = \{v, u\}$ *safe for* v.



- Start with any run of the greedy offline algorithm (which determines sample/reward status of the $2n$ values) such that:
  1. $e = \{v, u\}$ could be added to the **greedy solution w.r.t. samples**, *if* $V_{1,e} = 100$ *were a sample*
  2. $V_{1,e} = 100$ is a **reward**
- Fix a *small number* of coin flips to guarantee safety of e

Manipulating safe edges via greedy-ordered selection

Rewards in blue, samples in orange. Initially, all values are **anonymous**.
Goal: make $e = \{v, u\}$ *safe for* v.



- Start with any run of the greedy offline algorithm (which determines sample/reward status of the $2n$ values) such that:
  1. $e = \{v, u\}$ could be added to the **greedy solution w.r.t. samples**, *if* $V_{1,e} = 100$ *were a sample*
  2. $V_{1,e} = 100$ is a **reward**
- Fix a *small number* of coin flips to guarantee safety of e
- After fixing 2 coin flips, $e = \{v, u\}$ is safe for v (but *not* for u)

Proof for Matching with Edge Arrivals

With the safe edges defined, the competitive guarantee is (almost) immediate:

$$\mathbb{E}\left[w(\mathrm{ALG})\right] \geq \mathbb{E}\left[w(\text{Safe Edges})\right]$$

Can prove that the algorithm collects at least the weight of the safe edges

Proof for Matching with Edge Arrivals

With the safe edges defined, the competitive guarantee is (almost) immediate:

$$\begin{aligned}\mathbb{E}[w(\mathrm{ALG})] &\geq \mathbb{E}[w(\text{Safe Edges})] \\ &\geq {}^1/_8 \cdot \mathbb{E}[w(\text{Greedy Solution})]\end{aligned}$$

Loss to ensure edges are *safe*

Proof for Matching with Edge Arrivals

With the safe edges defined, the competitive guarantee is (almost) immediate:

$$\begin{aligned}\mathbb{E}\left[w(\mathrm{ALG})\right] &\geq \mathbb{E}\left[w(\text{Safe Edges})\right] \\ &\geq 1/8 \cdot \mathbb{E}\left[w(\text{Greedy Solution})\right] \\ &\geq 1/16 \cdot \mathbb{E}\left[w(\text{OPT})\right]\end{aligned}$$

Loss due to the greedy matching (2-approximation of OPT)

Reducing OOS to "Pointwise"-SSPIs

- Prophet inequalities appear "easier" than secretary problems:
  - $\exists$ a 2-approximation for matroid prophet inequalities
  - Best known (order-oblivious) matroid secretary is $\mathcal{O}(\log\log(\mathrm{rank}))$

Reducing OOS to “Pointwise”-SSPIs

- Prophet inequalities appear “easier” than secretary problems:
  - $\exists$ a 2-approximation for matroid prophet inequalities
  - Best known (order-oblivious) matroid secretary is $\mathcal{O}(\log\log(\mathrm{rank}))$
- What about SSPIs? Is there hope for a 2-competitive matroid SSPI?

Reducing OOS to "Pointwise"-SSPIs

Recall: Equivalent sample/reward generation for SSPIs

Viewpoint 1 (all offline):

1. Two samples $V_{1,e}, V_{2,e} \sim \mathcal{D}_e$ are drawn **independently** for every element e for *arbitrary* $\mathcal{D}_e$
2. A *single* independent coin flip for each e decides which is a sample/reward

Viewpoint 2:

1. Draw samples $S_e \sim D_e$ for each element e *offline*
2. Online, one at a time, draw reward $X_e \sim \mathcal{D}_e$

Reducing OOS to “Pointwise”-SSPIs

Definition (“Pointwise”-SSPI)

An SSPI which maintains its competitive guarantee when the rewards/samples are generated as follows:

1. Adversary chooses 2 **arbitrary** values $V_{1,e}, V_{2,e}$ for every element e
2. A *single* independent coin flip for each e decides which is a sample/reward

Reducing OOS to "Pointwise"-SSPIs

Definition ("Pointwise"-SSPI)

An SSPI which maintains its competitive guarantee when the rewards/samples are generated as follows:

1. Adversary chooses 2 **arbitrary** values $V_{1,e}, V_{2,e}$ for every element e
2. A *single* independent coin flip for each e decides which is a sample/reward

(*) Reward and sample can be *correlated*!!

Reducing OOS to "Pointwise"-SSPIs

Definition ("Pointwise"-SSPI)

An SSPI which maintains its competitive guarantee when the rewards/samples are generated as follows:

1. Adversary chooses 2 **arbitrary** values $V_{1,e}, V_{2,e}$ for every element e
2. A *single* independent coin flip for each e decides which is a sample/reward

(*) Reward and sample can be *correlated*!!

Observation

Every known SSPI is actually a "Pointwise"-SSPI

Reducing OOS to “Pointwise”-SSPIs

Definition (“Pointwise”-SSPI)

An SSPI which maintains its competitive guarantee when the rewards/samples are generated as follows:

1. Adversary chooses 2 **arbitrary** values $V_{1,e}, V_{2,e}$ for every element e
2. A *single* independent coin flip for each e decides which is a sample/reward

Theorem (“Pointwise”-SSPI implies OOS)

An α-competitive “Pointwise”-SSPI on any downward-closed feasible set implies an 2α-competitive OOS on the same feasible set

Reducing OOS to "Pointwise"-SSPIs

Definition ("Pointwise"-SSPI)

An SSPI which maintains its competitive guarantee when the rewards/samples are generated as follows:

1. Adversary chooses 2 **arbitrary** values $V_{1,e}, V_{2,e}$ for every element e
2. A *single* independent coin flip for each e decides which is a sample/reward

Theorem ("Pointwise"-SSPI implies OOS)

An α-competitive "Pointwise"-SSPI on any downward-closed feasible set implies an 2α-competitive OOS on the same feasible set

Theorem (Azar, Kleinberg, Weinberg '13: OOS implies SSPI)

An α-competitive OOS on any feasible set implies an α-competitive SSPI on the same feasible set

Reducing OOS to “Pointwise”-SSPIs

Theorem (“Pointwise”-SSPI implies OOS)

An α-competitive “Pointwise”-SSPI on any downward-closed feasible set implies an 2α-competitive OOS on the same feasible set

Theorem (Azar, Kleinberg, Weinberg ’13: OOS implies SSPI)

An α-competitive OOS on any feasible set implies an α-competitive SSPI on the same feasible set

- *Partial* converse to the reduction of AKW’13!

A closing question

Question

Are there SSPIs which are **not** *"Pointwise"??*

A closing question

Question

Are there SSPIs which are **not** *"Pointwise"??*

- If not, constant-competitive matroid SSPI would resolve *matroid secretary conjecture*...

A closing question

Question

Are there SSPIs which are **not** *"Pointwise"??*

- If not, constant-competitive matroid SSPI would resolve *matroid secretary conjecture*...

Thanks for listening!