Prophet Inequalities

Part 2: Online matching and contention resolution

Paul Dütting, Google Research

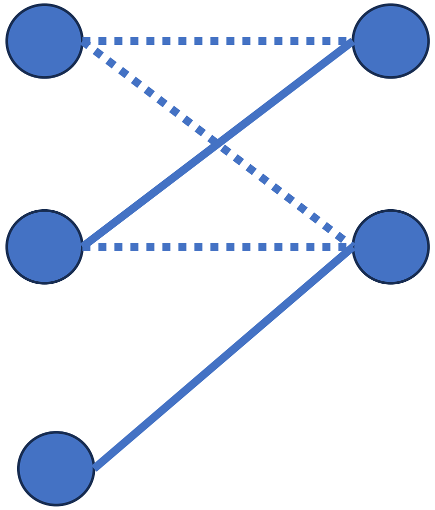
duetting@google.com

ADFOCS 2024 Summer School

August 2024

Online Matching

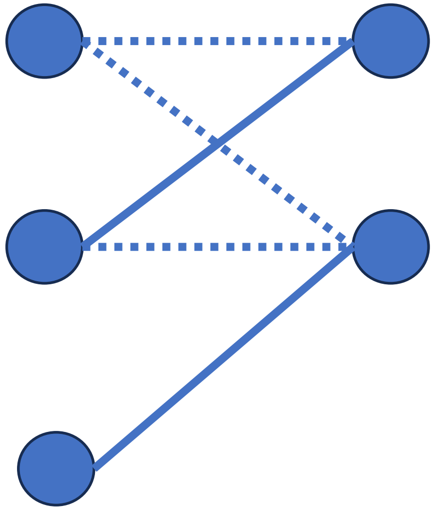
- Fundamental model with numerous applications:
    - Items and buyers (in e-commerce)
    - Drivers and passengers (in ridesharing platforms)
    - Ad slots and advertisers (in online ad auctions)
    - Jobs and workers (in online labor markets)
    - …



Example: Matching in bipartite graph

Online Matching

- Fundamental model with numerous applications:
  - Items and buyers (in e-commerce)
  - Drivers and passengers (in ridesharing platforms)
  - Ad slots and advertisers (in online ad auctions)
  - Jobs and workers (in online labor markets)
  - …



Example: Matching in bipartite graph

- Two flavors: edge arrival or vertex arrival (a.k.a. “batched arrival”)
- **Goal:** maximize total weight of chosen matching

Plan for Part 2

- Alternative proof for single-choice prophet inequality
  - via: “online contention resolution”
- Prophet inequalities for online matching via this technique
  - with edge arrivals in general graphs
  - with vertex (“batched”) arrivals in general graphs

High-Level Idea

- (Relax) Define a fractional relaxation
- (Round) Devise an online rounding scheme

Outline Other Parts

Part 1: Introduction

Part 2: Online matching and contention resolution

Part 3: Online combinatorial auctions and balanced prices

Part 4: Data-driven prophet inequalities

Additional References

Surveys and book chapters:

- “Online Matching: A Brief Survey” by Zhiyi Huang, Zhihao Gavin Tang, and David Wajc [SIGEcom ’24]
- “Applications of Online Matching” by Zhiyi Huang and Thorben Tröbst (Chapter 5 in Echenique/Immorlica/Vazirani book) [link]
- “Online Matching in Advertisement Auctions” by Nikhil Devanur and Aranyak Mehta (Chapter 6 in Echenique/Immorlica/Vazirani book) [link]
- “Online Matching and Ad Allocation” by Aranyak Mehta (FnT-TCS Survey) [link]

Additional References

Tutorials and workshops:

- FOCS 2023 Workshop “Online Algorithms and Online Rounding: Recent Progress” by Zhiyi Huang and David Wajc [website]
- WINE 2023 Tutorial “Recent Progress and Future Directions in Online Matching” by Zhiyi Huang and Zhihao Gavin Tang [slides-pt1, slides-pt2]

Recall: The Classic Prophet Inequality

The Problem

- Given known distributions $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_n$ over (non-negative) values:
  - A gambler gets to see realizations $v_i \sim \mathcal{D}_i$ one-by-one, and needs to immediately and irrevocable decide whether to accept v_i
  - The prophet sees the entire sequence of values $v_1, v_2, \dots, v_n$ at once, and can simply choose the maximum value
- **Question:** What's the worst-case gap between $\underbrace{\mathbb{E}[\text{value accepted by gambler}]}_{=:\,\mathbb{E}[ALG]}$ and $\underbrace{\mathbb{E}[\text{value accepted by prophet}]}_{=\,\mathbb{E}[\max_i v_i]}$?

Prophet Inequality

Theorem [Krengel-Succheston ‘77+’78] (+ Garling)

For all distributions $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n$, there is an algorithm ALG such that $\mathbb{E}[ALG] \geq \frac{1}{2}\, \mathbb{E}[\max_i v_i]$.

Krengel and Succheston in Oberwolfach

A Different Proof: Online Contention Resolution Scheme (OCRS)

[Chekuri Vondrak Zenklusen ‘14, Feldman Svensson Zenklusen ’16, Lee Singla ’18]

Proof via Contention Resolution

For simplicity suppose: $v_i = x_i$ with probability p_i, and $v_i = 0$ otherwise

Proof via Contention Resolution

For simplicity suppose: $v_i = x_i$ with probability p_i, and $v_i = 0$ otherwise

Lemma: $\underbrace{\mathbb{E}[\max_i v_i]}_{\text{PROPHET}}$ is at most:

$$\underbrace{\begin{array}{lll} \text{maximize} & \sum_{i=1}^{n} y_i \cdot x_i & \\ \text{subject to} & \sum_{i=1}^{n} y_i \leq 1 & \forall i \\ & y_i \in [0, p_i] & \forall i \end{array}}_{\text{“ex ante relaxation”}}$$

Proof via Contention Resolution

For simplicity suppose: $v_i = x_i$ with probability p_i, and $v_i = 0$ otherwise

Lemma: $\underbrace{\mathbb{E}[\max_i v_i]}_{\text{PROPHET}}$ is at most:

$$\underbrace{\begin{array}{lll} \text{maximize} & \sum_{i=1}^{n} y_i \cdot x_i & \\ \text{subject to} & \sum_{i=1}^{n} y_i \leq 1 & \forall\, i \\ & y_i \in [0, p_i] & \forall\, i \end{array}}_{\text{“ex ante relaxation”}}$$

Proof (of the lemma):

Setting $y_i = \Pr[x_i \text{ is chosen by prophet}]$ is feasible, and for this choice the objective value is equal to PROPHET.

Q.E.D.

Proof via Contention Resolution

Proof (of the prophet inequality):

ALG: Upon arrival of element i, if $v_i = x_i$, then pick element i with prob. $\frac{y_i}{2\,\alpha_i\, p_i}$,

where $\alpha_i = 1 - \frac{1}{2}\sum_{j<i} y_j$ (= probability element i is reached)

Proof via Contention Resolution

Proof (of the prophet inequality):

ALG: Upon arrival of element i, if $v_i = x_i$, then pick element i with prob. $\frac{y_i}{2\,\alpha_i\,p_i}$,

where $\alpha_i = 1 - \frac{1}{2}\sum_{j<i} y_j$ (= probability element i is reached)

Analysis:

- $\frac{y_i}{2\,\alpha_i\,p_i} \le \frac{1}{2\,\alpha_i} \le 1$ where we used (1) $y_i \le p_i$ and (2) $\alpha_i \ge 1 - \frac{1}{2}\sum_j y_j \ge \frac{1}{2}$

Proof via Contention Resolution

Proof (of the prophet inequality):

ALG: Upon arrival of element i, if $v_i = x_i$, then pick element i with prob. $\frac{y_i}{2\,\alpha_i\,p_i}$,

where $\alpha_i = 1 - \frac{1}{2}\sum_{j<i} y_j$ (= probability element i is reached)

Analysis:

- $\frac{y_i}{2\,\alpha_i\,p_i} \leq \frac{1}{2\,\alpha_i} \leq 1$ where we used (1) $y_i \leq p_i$ and (2) $\alpha_i \geq 1 - \frac{1}{2}\sum_j y_j \geq \frac{1}{2}$
- Every element i is picked w.p. $y_i/2$. Proof by induction:
  - Suppose this holds for every element $j < i$
  - Then $\Pr[i \text{ is picked}] = \Pr[i \text{ is reached}] \cdot \Pr[v_i = x_i] \cdot \frac{y_i}{2\,\alpha_i\,p_i} = \alpha_i \cdot p_i \cdot \frac{y_i}{2\,\alpha_i\,p_i} = \frac{y_i}{2}$

Proof via Contention Resolution

Proof (of the prophet inequality):

ALG: Upon arrival of element i, if $v_i = x_i$, then pick element i with prob. $\frac{y_i}{2\,\alpha_i\,p_i}$,

where $\alpha_i = 1 - \frac{1}{2}\sum_{j<i} y_j$ (= probability element i is reached)

Analysis:

- $\frac{y_i}{2\,\alpha_i\,p_i} \le \frac{1}{2\,\alpha_i} \le 1$ where we used (1) $y_i \le p_i$ and (2) $\alpha_i \ge 1 - \frac{1}{2}\sum_j y_j \ge \frac{1}{2}$
- Every element i is picked w.p. $y_i/2$. Proof by induction:
  - Suppose this holds for every element $j < i$
  - Then $\Pr[i \text{ is picked}] = \Pr[i \text{ is reached}] \cdot \Pr[v_i = x_i] \cdot \frac{y_i}{2\,\alpha_i\,p_i} = \alpha_i \cdot p_i \cdot \frac{y_i}{2\,\alpha_i\,p_i} = \frac{y_i}{2}$
- And so $\mathbb{E}[\mathrm{ALG}] = \sum_{i=1}^{n} \frac{y_i}{2}\, x_i \ge \frac{1}{2}\mathbb{E}[\max_i v_i]$ (by lemma)

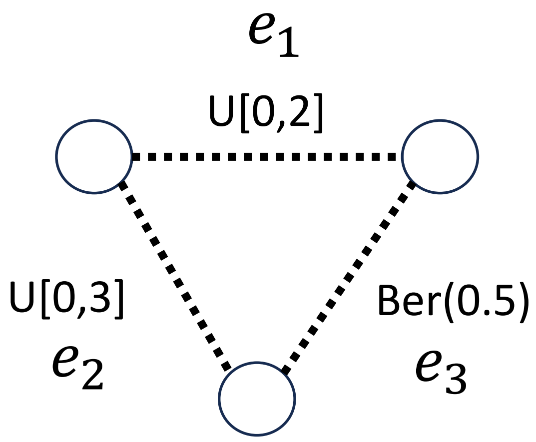
Q.E.D.

Online Matching with Edge Arrivals (in general graphs)

[Gravin Wang ‘19, Ezra Feldman Gravin Tang ’20, MacRury Ma Grammel ‘23]

Matching with Edge Arrivals

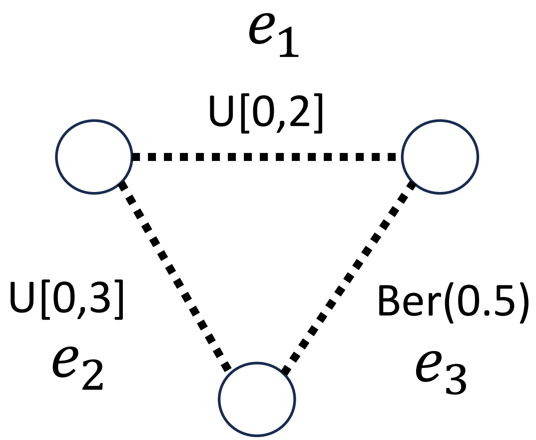
- A weighted graph $G = (V, E)$ (not necessarily bipartite)
- Edge e has weight $w_e \sim \mathcal{D}_e$
  - Initially: w_e unknown, $\mathcal{D}_e$ known
- Upon arrival of an edge e, its weight w_e is revealed
- ALG decides whether to include e in the matching (if feasible)



Matching with Edge Arrivals

- A weighted graph $G = (V, E)$ (not necessarily bipartite)
- Edge e has weight $w_e \sim \mathcal{D}_e$
  - Initially: w_e unknown, $\mathcal{D}_e$ known
- Upon arrival of an edge e, its weight w_e is revealed
- ALG decides whether to include e in the matching (if feasible)

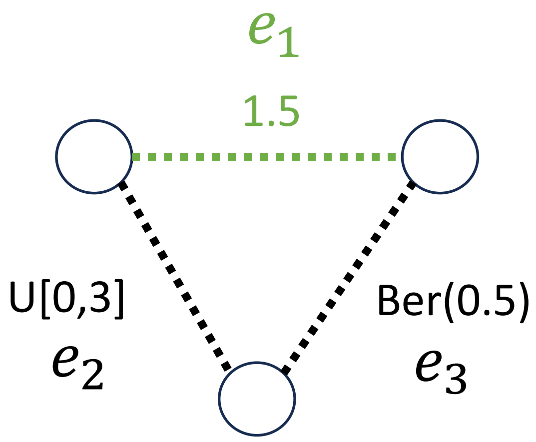
- **Goal:** Maximize expected total weight
- **Benchmark (“prophet”):** Expected weight of offline optimum



Matching with Edge Arrivals

- A weighted graph $G = (V, E)$ (not necessarily bipartite)
- Edge e has weight $w_e \sim \mathcal{D}_e$
  - Initially: w_e unknown, $\mathcal{D}_e$ known
- Upon arrival of an edge e, its weight w_e is revealed
- ALG decides whether to include e in the matching (if feasible)

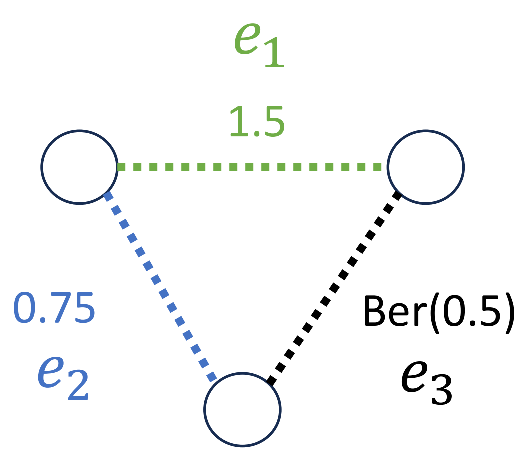
- **Goal:** Maximize expected total weight
- **Benchmark ("prophet"):** Expected weight of offline optimum



Matching with Edge Arrivals

- A weighted graph $G = (V, E)$ (not necessarily bipartite)
- Edge e has weight $w_e \sim \mathcal{D}_e$
  - Initially: w_e unknown, $\mathcal{D}_e$ known
- Upon arrival of an edge e, its weight w_e is revealed
- ALG decides whether to include e in the matching (if feasible)

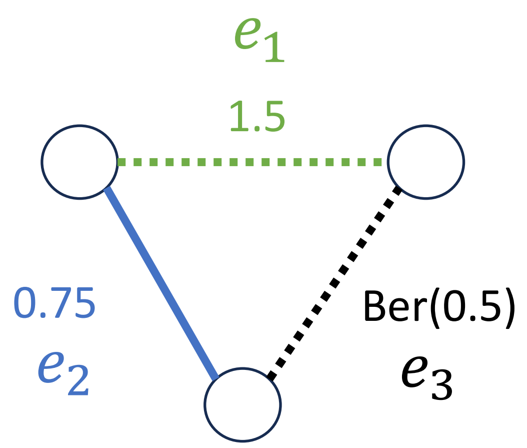
- **Goal:** Maximize expected total weight
- **Benchmark ("prophet"):** Expected weight of offline optimum



Matching with Edge Arrivals

- A weighted graph $G = (V, E)$ (not necessarily bipartite)
- Edge e has weight $w_e \sim \mathcal{D}_e$
  - Initially: w_e unknown, $\mathcal{D}_e$ known
- Upon arrival of an edge e, its weight w_e is revealed
- ALG decides whether to include e in the matching (if feasible)

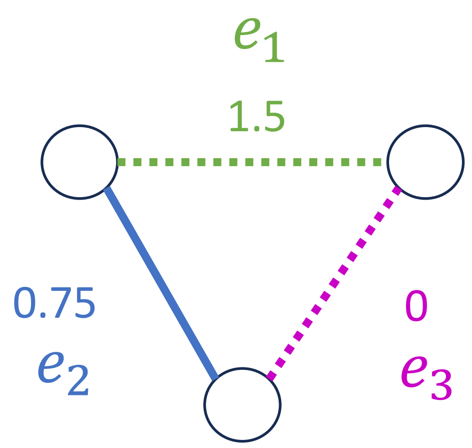
- **Goal:** Maximize expected total weight
- **Benchmark ("prophet"):** Expected weight of offline optimum



Matching with Edge Arrivals

- A weighted graph $G = (V, E)$ (not necessarily bipartite)
- Edge e has weight $w_e \sim \mathcal{D}_e$
  - Initially: w_e unknown, $\mathcal{D}_e$ known
- Upon arrival of an edge e, its weight w_e is revealed
- ALG decides whether to include e in the matching (if feasible)

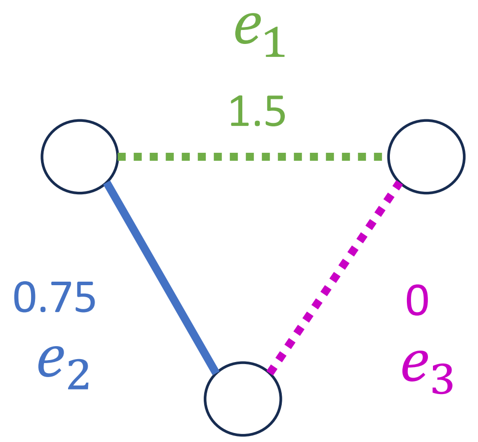
- **Goal:** Maximize expected total weight
- **Benchmark ("prophet"):** Expected weight of offline optimum



Matching with Edge Arrivals

- A weighted graph $G = (V, E)$ (not necessarily bipartite)
- Edge e has weight $w_e \sim \mathcal{D}_e$
  - Initially: w_e unknown, $\mathcal{D}_e$ known
- Upon arrival of an edge e, its weight w_e is revealed
- ALG decides whether to include e in the matching (if feasible)

- **Goal:** Maximize expected total weight
- **Benchmark ("prophet"):** Expected weight of offline optimum



ALG = 0.75
OPT = 1.5

Prophet Inequality

Theorem [Ezra Feldman Gravin Tang '20]

There is an algorithm for online matching with edge arrivals in general graphs that is 3-competitive against the prophet benchmark.

Prophet Inequality

Theorem [Ezra Feldman Gravin Tang '20]

There is an algorithm for online matching with edge arrivals in general graphs that is 3-competitive against the prophet benchmark.

State of the art:

	Lower bound	**Upper bound**
Bipartite graphs	≥ 2.25 [Gravin and Wang '19] $\geq 7/3$ [Correa Cristi Fielbaum Pollner Weinberg '22]	≤ 3 [Gravin Tang '19] ≤ 2.865 [MacRury Ma Grammel '23]
General graphs	≥ 2.5 [MacRury, Ma, Grammel '23] ≥ 2.564 for OCRS-based approaches	≤ 2.967 [Ezra Feldman Gravin Tang '20] ≤ 2.907 [MacRury Ma Grammel '23]

OCRS Proof

For simplicity suppose: $w_e = x_e$ with probability p_e, and $w_e = 0$ otherwise

Lemma: $\underbrace{\mathbb{E}\left[\max_{\mathrm{M}} \mathrm{w}(\mathrm{M})\right]}_{\text{PROPHET}}$ is at most:

$$\underbrace{\begin{array}{lll} \max & \sum_e y_e \cdot x_e & \\ \text{s.t.} & \sum_{e:u\in e} y_e \le 1 & \forall \text{ node } u \\ & y_e \in [0, p_e] & \forall \text{ edge } e \end{array}}_{\text{"ex ante relaxation"}}$$

(cf. fractional matching polytope)

OCRS Proof

Proof (of the prophet inequality):

ALG: Upon arrival of edge $e = (u, v)$, if $w_e = x_e$ and e is available (i.e., u and v are available), then match edge e with prob. $\frac{y_e}{3\,\alpha_e\,p_e}$, where $\alpha_e = \Pr[e \text{ available}]$

OCRS Proof

Proof (of the prophet inequality):

ALG: Upon arrival of edge $e = (u, v)$, if $w_e = x_e$ and e is available (i.e., u and v are available), then match edge e with prob. $\frac{y_e}{3\,\alpha_e\,p_e}$, where $\alpha_e = \Pr[e \text{ available}]$

Analysis:

- $\Pr[e \text{ matched}] = \Pr[e \text{ available}] \cdot \Pr[w_e = x_e] \cdot \frac{y_e}{3\,\alpha_e\,p_e} = \alpha_e \cdot p_e \cdot \frac{y_e}{3\,\alpha_e\,p_e} = \frac{y_e}{3}$

OCRS Proof

Proof (of the prophet inequality):

ALG: Upon arrival of edge $e = (u, v)$, if $w_e = x_e$ and e is available (i.e., u and v are available), then match edge e with prob. $\frac{y_e}{3\,\alpha_e\,p_e}$, where $\alpha_e = \Pr[e \text{ available}]$

Analysis:

- $\Pr[e \text{ matched}] = \Pr[e \text{ available}] \cdot \Pr[w_e = x_e] \cdot \frac{y_e}{3\,\alpha_e\,p_e} = \alpha_e \cdot p_e \cdot \frac{y_e}{3\,\alpha_e\,p_e} = \frac{y_e}{3}$
- $\frac{y_e}{3\,\alpha_e\,p_e} \leq \frac{1}{3\,\alpha_e} \leq 1$ because (1) $y_e \leq p_e$ and (2) $\alpha_e \geq 1 - \Pr[u \text{ unvailable}] - \Pr[v \text{ unavailable}] \geq \frac{1}{3}$ (by union bound)
  - $\Pr[u \text{ unvailable}] = \sum_{e' < e : u \in e'} \Pr[e' \text{matched}] = \sum_{e' < e : u \in e'} \frac{y_{e'}}{3} \leq \frac{1}{3}$

OCRS Proof

Proof (of the prophet inequality):

ALG: Upon arrival of edge $e = (u, v)$, if $w_e = x_e$ and e is available (i.e., u and v are available), then match edge e with prob. $\frac{y_e}{3\,\alpha_e\,p_e}$, where $\alpha_e = \Pr[e \text{ available}]$

Analysis:

- $\Pr[e \text{ matched}] = \Pr[e \text{ available}] \cdot \Pr[w_e = x_e] \cdot \frac{y_e}{3\,\alpha_e\,p_e} = \alpha_e \cdot p_e \cdot \frac{y_e}{3\,\alpha_e\,p_e} = \frac{y_e}{3}$
- $\frac{y_e}{3\,\alpha_e\,p_e} \leq \frac{1}{3\,\alpha_e} \leq 1$ because (1) $y_e \leq p_e$ and (2) $\alpha_e \geq 1 - \Pr[u \text{ unvailable}] - \Pr[v \text{ unavailable}] \geq \frac{1}{3}$ (by union bound)
  - $\Pr[u \text{ unvailable}] = \sum_{e' < e : u \in e'} \Pr[e' \text{matched}] = \sum_{e' < e : u \in e'} \frac{y_{e'}}{3} \leq \frac{1}{3}$
- And so $\mathbb{E}[\text{ALG}] = \sum_e \frac{y_e}{3}\, x_e \geq \frac{1}{3}\,\mathbb{E}\left[\max_{\text{M}} \text{w}(\text{M})\right]$ (by lemma)

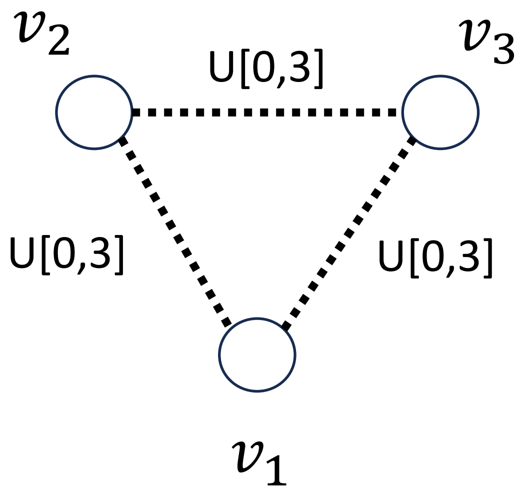
Q.E.D.

Online Matching with Vertex Arrivals (in general graphs)

[Ezra Feldman Gravin Tang ‘20]

Matching with Vertex Arrivals

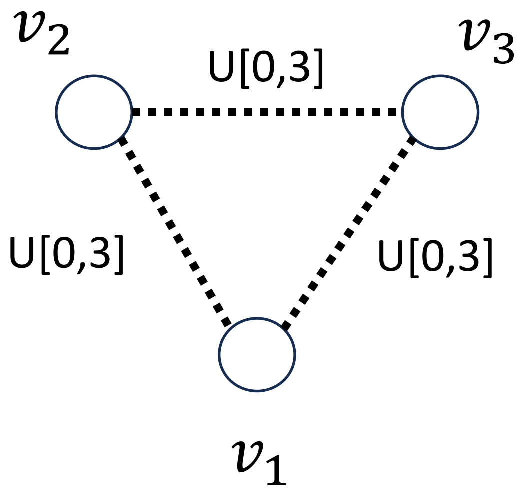
- A weighted graph $G = (V, E)$ (not necessarily bipartite)
- Edge e has weight $w_e \sim \mathcal{D}_e$
  - Initially: w_e unknown, $\mathcal{D}_e$ known
- Upon arrival of a vertex $v \in V$, weights of edges to previously arrived vertices are revealed
- ALG decides to whom vertex v is matched (if at all)



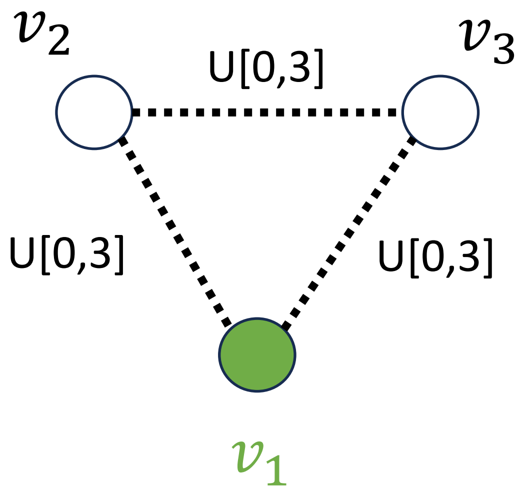
Matching with Vertex Arrivals

- A weighted graph $G = (V, E)$ (not necessarily bipartite)
- Edge e has weight $w_e \sim \mathcal{D}_e$
  - Initially: w_e unknown, $\mathcal{D}_e$ known
- Upon arrival of a vertex $v \in V$, weights of edges to previously arrived vertices are revealed
- ALG decides to whom vertex v is matched (if at all)

- **Goal:** Maximize expected total weight
- **Benchmark ("prophet"):** Expected weight of offline optimum



Matching with Vertex Arrivals



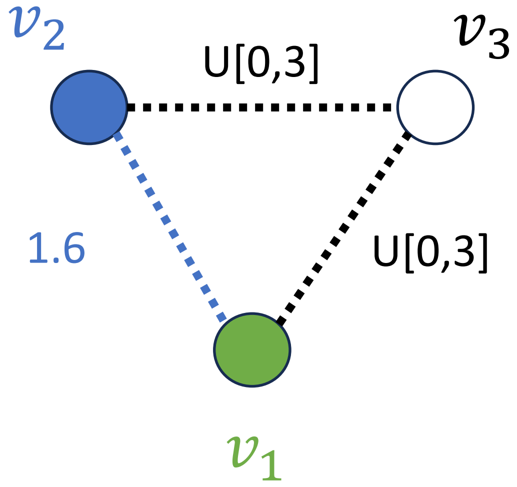
- A weighted graph $G = (V, E)$ (not necessarily bipartite)
- Edge e has weight $w_e \sim \mathcal{D}_e$
  - Initially: w_e unknown, $\mathcal{D}_e$ known
- Upon arrival of a vertex $v \in V$, weights of edges to previously arrived vertices are revealed
- ALG decides to whom vertex v is matched (if at all)

- **Goal:** Maximize expected total weight
- **Benchmark ("prophet"):** Expected weight of offline optimum

Matching with Vertex Arrivals

- A weighted graph $G = (V, E)$ (not necessarily bipartite)
- Edge e has weight $w_e \sim \mathcal{D}_e$
  - Initially: w_e unknown, $\mathcal{D}_e$ known
- Upon arrival of a vertex $v \in V$, weights of edges to previously arrived vertices are revealed
- ALG decides to whom vertex v is matched (if at all)

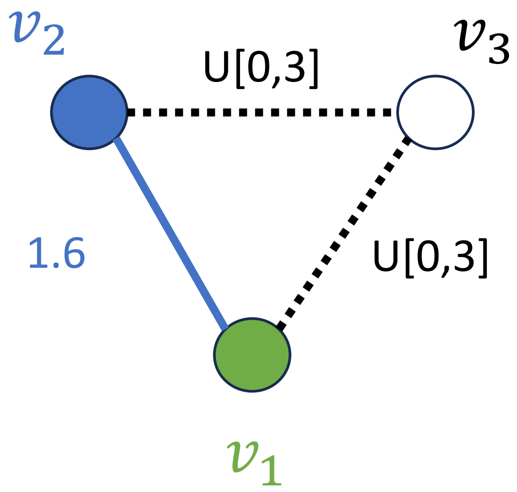
- **Goal:** Maximize expected total weight
- **Benchmark ("prophet"):** Expected weight of offline optimum



Matching with Vertex Arrivals

- A weighted graph $G = (V, E)$ (not necessarily bipartite)
- Edge e has weight $w_e \sim \mathcal{D}_e$
  - Initially: w_e unknown, $\mathcal{D}_e$ known
- Upon arrival of a vertex $v \in V$, weights of edges to previously arrived vertices are revealed
- ALG decides to whom vertex v is matched (if at all)

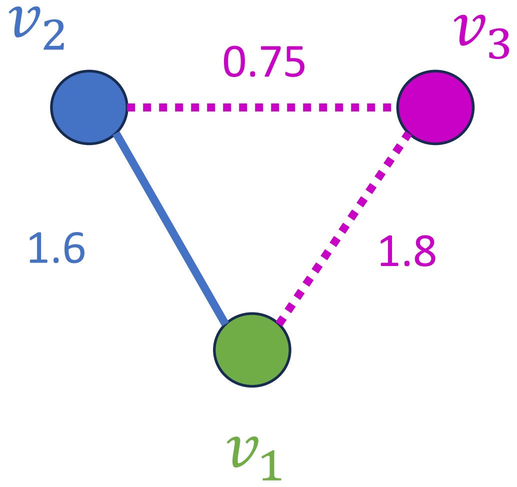
- **Goal:** Maximize expected total weight
- **Benchmark ("prophet"):** Expected weight of offline optimum



Matching with Vertex Arrivals

- A weighted graph $G = (V, E)$ (not necessarily bipartite)
- Edge e has weight $w_e \sim \mathcal{D}_e$
  - Initially: w_e unknown, $\mathcal{D}_e$ known
- Upon arrival of a vertex $v \in V$, weights of edges to previously arrived vertices are revealed
- ALG decides to whom vertex v is matched (if at all)

- **Goal:** Maximize expected total weight
- **Benchmark ("prophet"):** Expected weight of offline optimum



ALG = 1.6
OPT = 1.8

Prophet Inequality

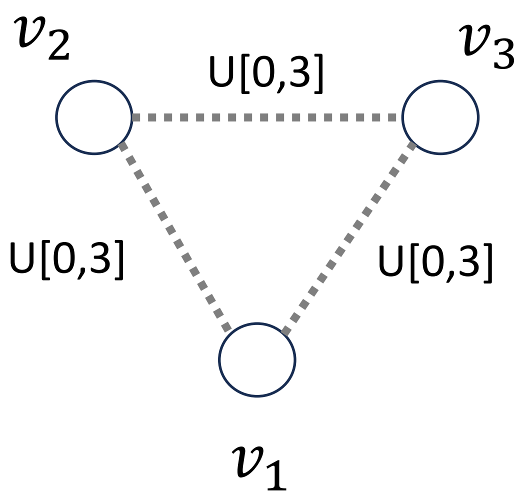
Theorem [Ezra Feldman Gravin Tang '20]

There is an algorithm for online matching with vertex arrivals in general graphs that is 2-competitive against the prophet benchmark.

(this is best possible)

Algorithm for Vertex Arrivals

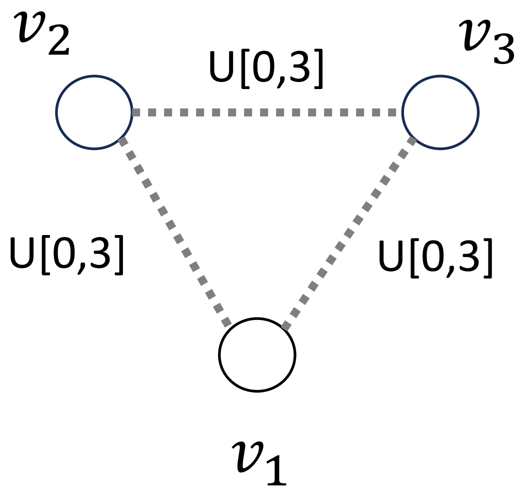
- Precompute: $y_e = \Pr[e \in OPT]$



Algorithm for Vertex Arrivals

- Precompute: $y_e = \Pr[e \in OPT]$

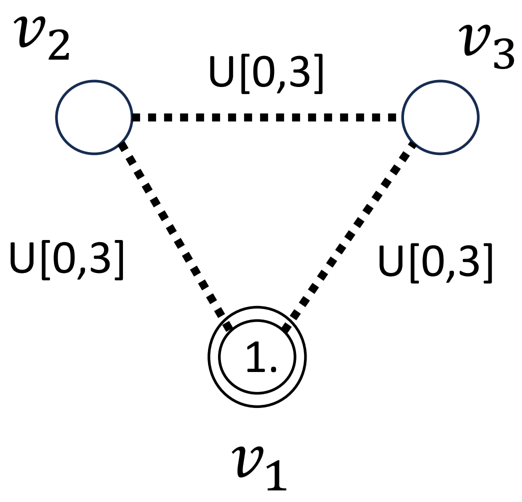
$y_e = \frac{1}{3}$ for all $e \in E$



Algorithm for Vertex Arrivals

- Precompute: $y_e = \Pr[e \in OPT]$
- Upon arrival of vertex v:
  - B_v ≔ edges from v to former vertices; $-B_v \coloneqq E \backslash \mathrm{B}_v$

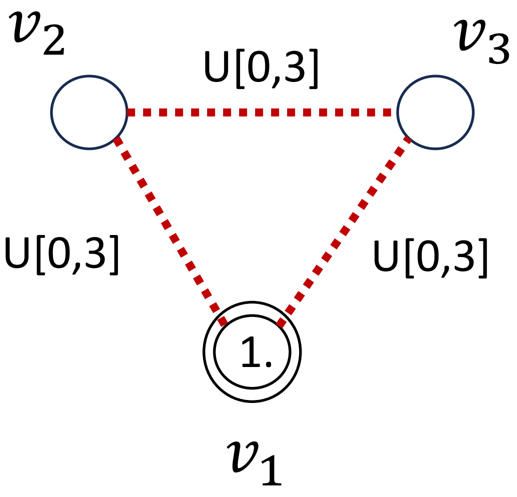
$y_e = \frac{1}{3}$ for all $e \in E$



Algorithm for Vertex Arrivals

- Precompute: $y_e = \Pr[e \in OPT]$
- Upon arrival of vertex v:
  - B_v ≔ edges from v to former vertices; $-B_v \coloneqq E \backslash \mathrm{B}_v$

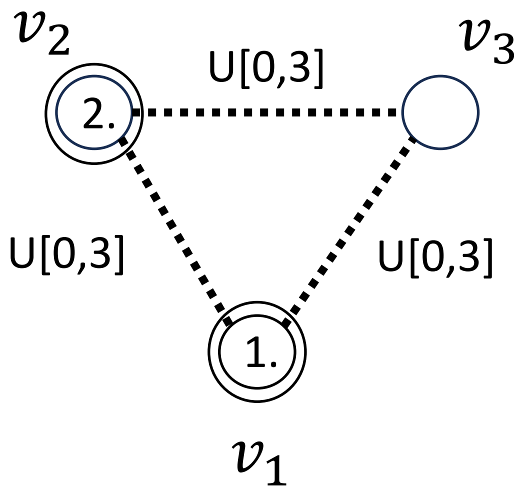
$y_e = \frac{1}{3}$ for all $e \in E$



Algorithm for Vertex Arrivals

- Precompute: $y_e = \Pr[e \in OPT]$
- Upon arrival of vertex v:
  - B_v ≔ edges from v to former vertices; $-B_v \coloneqq E \setminus \mathrm{B}_v$

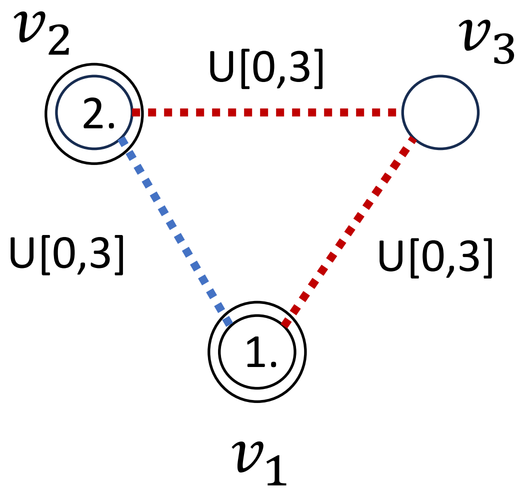
$y_e = \frac{1}{3}$ for all $e \in E$



Algorithm for Vertex Arrivals

- Precompute: $y_e = \Pr[e \in OPT]$
- Upon arrival of vertex v:
  - B_v ≔ edges from v to former vertices; $-B_v \coloneqq E \backslash B_v$

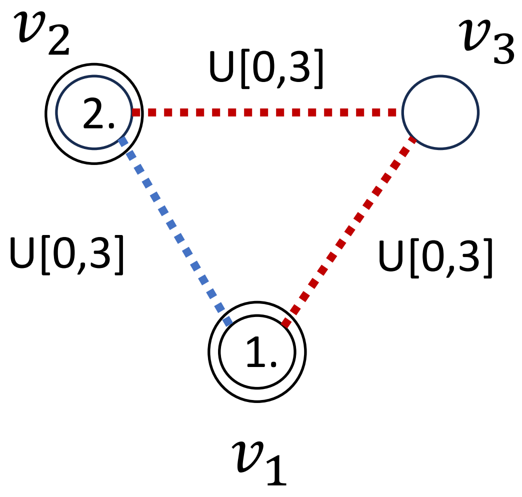
$y_e = \frac{1}{3}$ for all $e \in E$



Algorithm for Vertex Arrivals

- Precompute: $y_e = \Pr[e \in OPT]$
- Upon arrival of vertex v:
  - B_v ≔ edges from v to former vertices; $-B_v \coloneqq E \backslash \mathrm{B}_v$
  - Observe weights w_e of new edges B_v

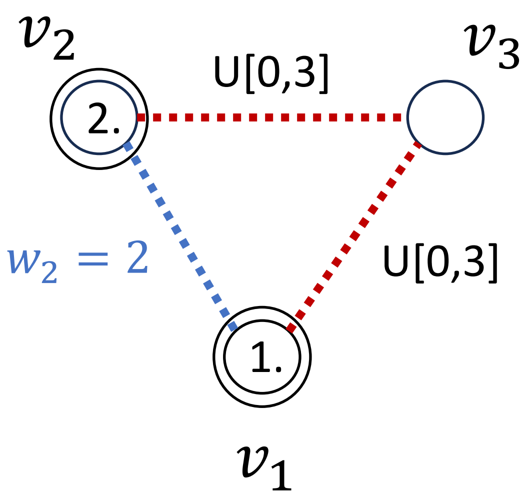
$y_e = \frac{1}{3}$ for all $e \in E$



Algorithm for Vertex Arrivals

- Precompute: $y_e = \Pr[e \in OPT]$
- Upon arrival of vertex v:
  - B_v ≔ edges from v to former vertices; $-B_v \coloneqq E \setminus B_v$
  - Observe weights w_e of new edges B_v

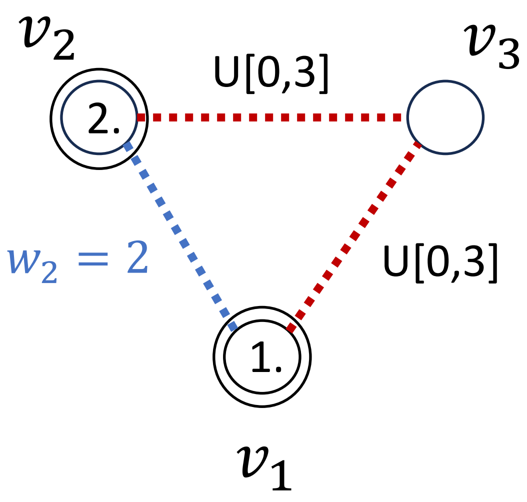
$y_e = \frac{1}{3}$ for all $e \in E$



Algorithm for Vertex Arrivals

- Precompute: $y_e = \Pr[e \in OPT]$
- Upon arrival of vertex v:
  - B_v ≔ edges from v to former vertices; $-B_v ≔ E \backslash \mathrm{B}_v$
  - Observe weights w_e of new edges B_v
  - Sample weights $\tilde{w}_{e'}$ for edges $e' \in -B_v$

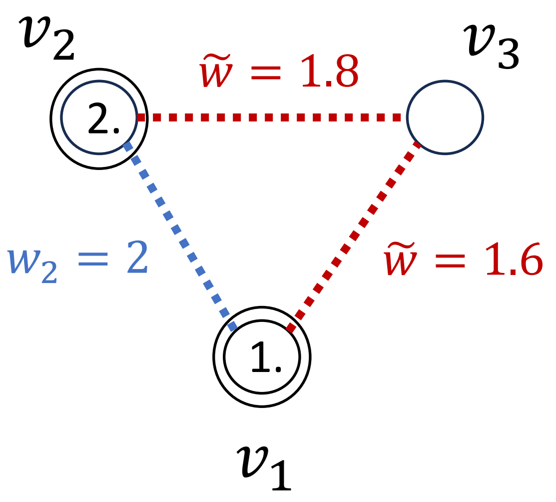
$y_e = \frac{1}{3}$ for all $e \in E$



Algorithm for Vertex Arrivals

- Precompute: $y_e = \Pr[e \in OPT]$
- Upon arrival of vertex v:
  - B_v ≔ edges from v to former vertices; $-B_v \coloneqq E \backslash \mathrm{B}_v$
  - Observe weights w_e of new edges B_v
  - Sample weights $\widetilde{w}_{e'}$ for edges $e' \in -B_v$

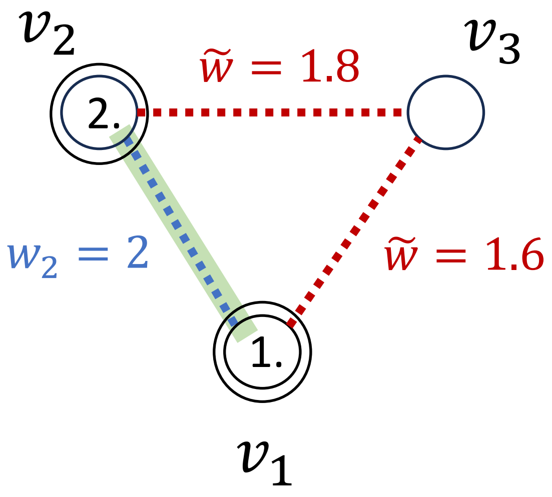
$y_e = \frac{1}{3}$ for all $e \in E$



Algorithm for Vertex Arrivals

- Precompute: $y_e = \Pr[e \in OPT]$
- Upon arrival of vertex v:
  - B_v ≔ edges from v to former vertices; $-B_v \coloneqq E \setminus B_v$
  - Observe weights w_e of new edges B_v
  - Sample weights $\tilde{w}_{e'}$ for edges $e' \in -B_v$
  - u ≔ partner of v in max-weight matching on $(w_{B_v}, \tilde{w}_{-B_v})$

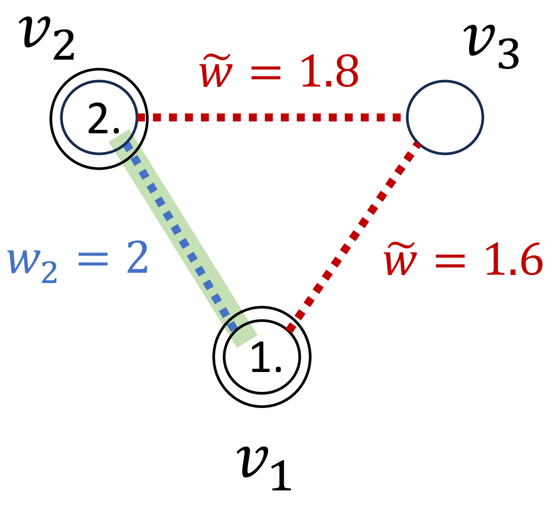
$y_e = \frac{1}{3}$ for all $e \in E$



Algorithm for Vertex Arrivals

- Precompute: $y_e = \Pr[e \in OPT]$
- Upon arrival of vertex v:
  - B_v ≔ edges from v to former vertices; $-B_v \coloneqq E \backslash \mathrm{B}_v$
  - Observe weights w_e of new edges B_v
  - Sample weights $\tilde{w}_{e'}$ for edges $e' \in -B_v$
  - u ≔ partner of v in max-weight matching on $\left(w_{B_v}, \tilde{w}_{-B_v}\right)$
  - If (a) u exists, (b) $u < v$, (c) u is available, then
    - Match v to u with probability $\alpha_u(v) = \frac{1}{2 - \sum_{r<v} y_{(r,u)}}$

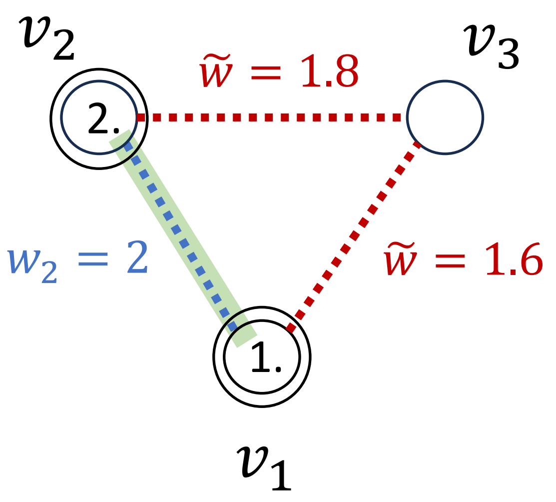
$y_e = \frac{1}{3}$ for all $e \in E$



Algorithm for Vertex Arrivals

- Precompute: $y_e = \Pr[e \in OPT]$
- Upon arrival of vertex v:
  - B_v := edges from v to former vertices; $-B_v := E \backslash \mathrm{B}_v$
  - Observe weights w_e of new edges B_v
  - Sample weights $\widetilde{w}_{e'}$ for edges $e' \in -B_v$
  - u := partner of v in max-weight matching on $\left(w_{B_v}, \widetilde{w}_{-B_v}\right)$
  - If (a) u exists, (b) $u < v$, (c) u is available, then
    - Match v to u with probability $\alpha_u(v) = \frac{1}{2 - \sum_{r<v} y_{(r,u)}}$

$y_e = \frac{1}{3}$ for all $e \in E$

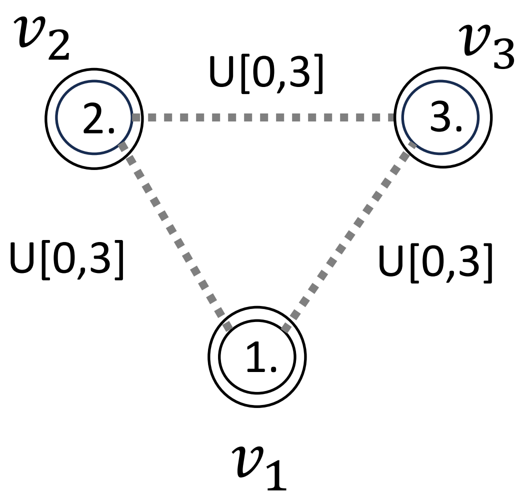


Match (2, 1) with prob. $\frac{1}{2 - \sum_{r<2} y_{(r,1)}} = \frac{1}{2}$

Algorithm for Vertex Arrivals

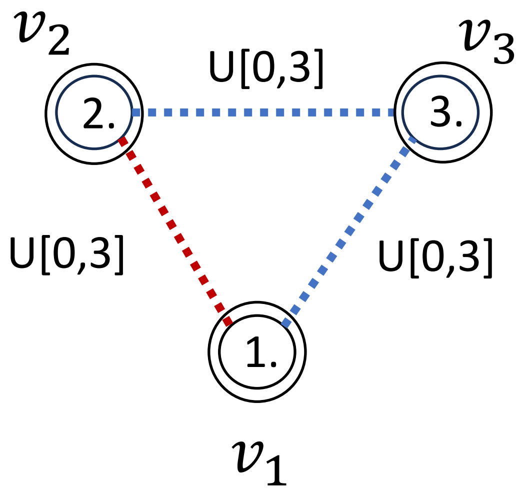
- Precompute: $y_e = \Pr[e \in OPT]$
- Upon arrival of vertex v:
  - B_v ≔ edges from v to former vertices; $-B_v \coloneqq E \setminus B_v$
  - Observe weights w_e of new edges B_v
  - Sample weights $\widetilde{w}_{e'}$ for edges $e' \in -B_v$
  - u ≔ partner of v in max-weight matching on $\left(w_{B_v}, \widetilde{w}_{-B_v}\right)$
  - If (a) u exists, (b) $u < v$, (c) u is available, then
    - Match v to u with probability $\alpha_u(v) = \frac{1}{2 - \sum_{r<v} y_{(r,u)}}$

$y_e = \frac{1}{3}$ for all $e \in E$



Algorithm for Vertex Arrivals

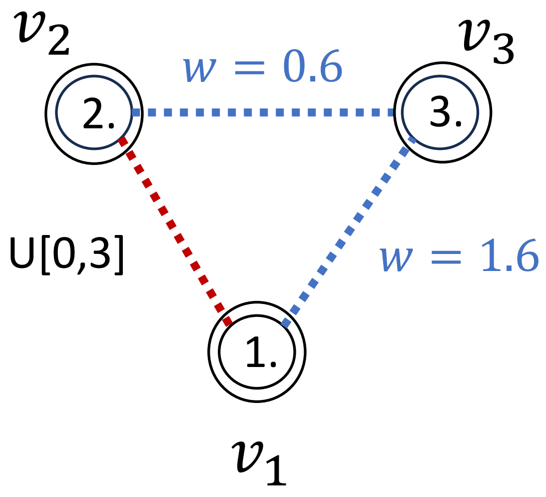
$y_e = \frac{1}{3}$ for all $e \in E$



- Precompute: $y_e = \Pr[e \in OPT]$
- Upon arrival of vertex v:
  - B_v ≔ edges from v to former vertices; $-B_v \coloneqq E \backslash \mathrm{B}_v$
  - Observe weights w_e of new edges B_v
  - Sample weights $\widetilde{w}_{e'}$ for edges $e' \in -B_v$
  - u ≔ partner of v in max-weight matching on $\left(w_{B_v}, \widetilde{w}_{-B_v}\right)$
  - If (a) u exists, (b) $u < v$, (c) u is available, then
    - Match v to u with probability $\alpha_u(v) = \frac{1}{2-\sum_{r<v} y_{(r,u)}}$

Algorithm for Vertex Arrivals

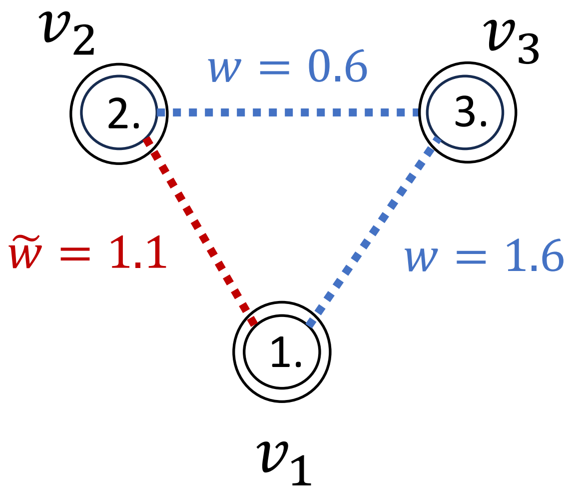
$y_e = \frac{1}{3}$ for all $e \in E$



- Precompute: $y_e = \Pr[e \in OPT]$
- Upon arrival of vertex v:
  - B_v ≔ edges from v to former vertices; $-B_v \coloneqq E \backslash \mathrm{B}_v$
  - Observe weights w_e of new edges B_v
  - Sample weights $\widetilde{w}_{e'}$ for edges $e' \in -B_v$
  - u ≔ partner of v in max-weight matching on $\left(w_{B_v}, \widetilde{w}_{-B_v}\right)$
  - If (a) u exists, (b) $u < v$, (c) u is available, then
    - Match v to u with probability $\alpha_u(v) = \frac{1}{2 - \sum_{r<v} y_{(r,u)}}$

Algorithm for Vertex Arrivals

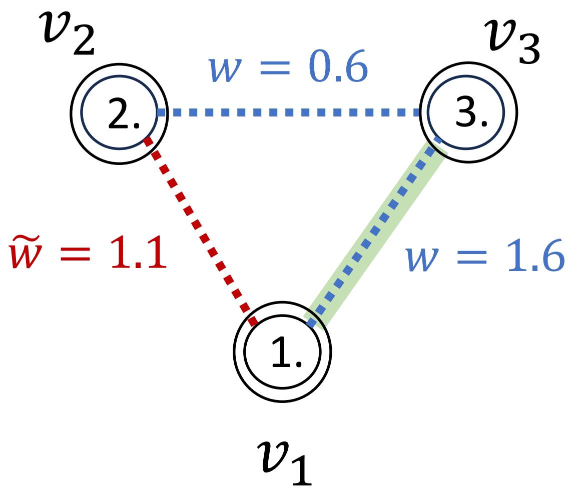
$y_e = \frac{1}{3}$ for all $e \in E$



- Precompute: $y_e = \Pr[e \in OPT]$
- Upon arrival of vertex v:
  - B_v ≔ edges from v to former vertices; $-B_v \coloneqq E \backslash B_v$
  - Observe weights w_e of new edges B_v
  - Sample weights $\tilde{w}_{e'}$ for edges $e' \in -B_v$
  - u ≔ partner of v in max-weight matching on $\left(w_{B_v}, \tilde{w}_{-B_v}\right)$
  - If (a) u exists, (b) $u < v$, (c) u is available, then
    - Match v to u with probability $\alpha_u(v) = \frac{1}{2 - \sum_{r<v} y_{(r,u)}}$

Algorithm for Vertex Arrivals

$y_e = \frac{1}{3}$ for all $e \in E$

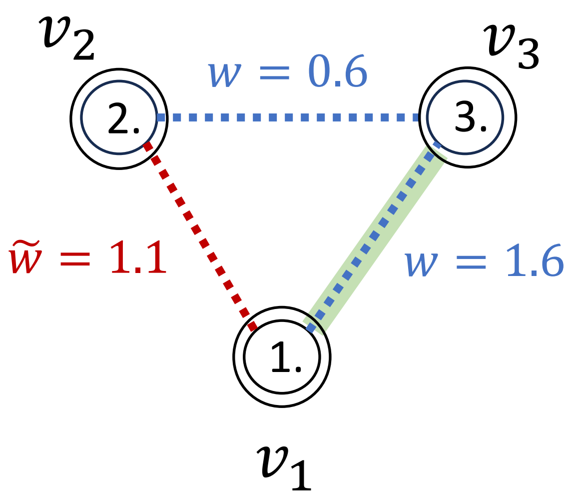


- Precompute: $y_e = \Pr[e \in OPT]$
- Upon arrival of vertex v:
  - B_v ≔ edges from v to former vertices; $-B_v \coloneqq E \backslash \mathrm{B}_v$
  - Observe weights w_e of new edges B_v
  - Sample weights $\tilde{w}_{e'}$ for edges $e' \in -B_v$
  - u ≔ partner of v in max-weight matching on $\left(w_{B_v}, \tilde{w}_{-B_v}\right)$
  - If (a) u exists, (b) $u < v$, (c) u is available, then
    - Match v to u with probability $\alpha_u(v) = \frac{1}{2 - \sum_{r<v} y_{(r,u)}}$

Algorithm for Vertex Arrivals

- Precompute: $y_e = \Pr[e \in OPT]$
- Upon arrival of vertex v:
  - $B_v :=$ edges from v to former vertices; $-B_v := E \backslash B_v$
  - Observe weights w_e of new edges B_v
  - Sample weights $\tilde{w}_{e'}$ for edges $e' \in -B_v$
  - $u :=$ partner of v in max-weight matching on $\left(w_{B_v}, \tilde{w}_{-B_v}\right)$
  - If (a) u exists, (b) $u < v$, (c) u is available, then
    - Match v to u with probability $\alpha_u(v) = \frac{1}{2 - \sum_{r<v} y_{(r,u)}}$

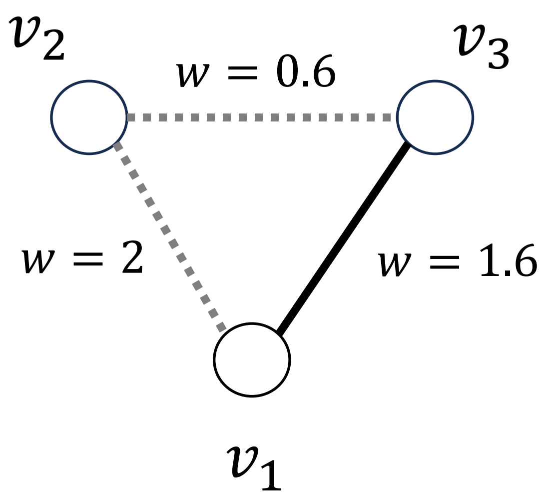
$y_e = \frac{1}{3}$ for all $e \in E$



Match (3, 1) with prob. $\frac{1}{2 - \sum_{r<3} y_{(r,1)}} = \frac{3}{5}$

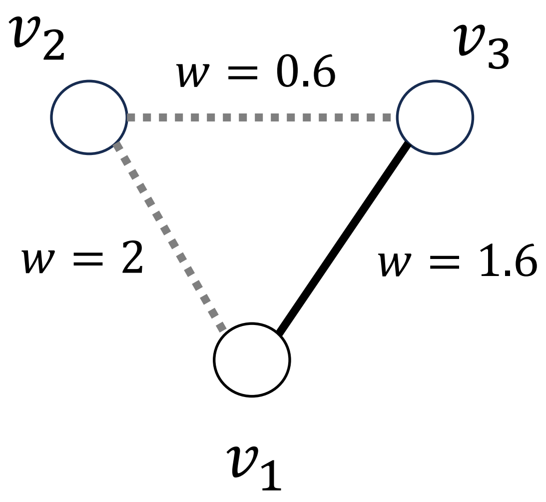
Algorithm for Vertex Arrivals

- Precompute: $y_e = \Pr[e \in OPT]$
- Upon arrival of vertex v:
  - B_v ≔ edges from v to former vertices; $-B_v \coloneqq E \backslash \mathrm{B}_v$
  - Observe weights w_e of new edges B_v
  - Sample weights $\tilde{w}_{e'}$ for edges $e' \in -B_v$
  - u ≔ partner of v in max-weight matching on $\left(w_{B_v}, \tilde{w}_{-B_v}\right)$
  - If (a) u exists, (b) $u < v$, (c) u is available, then
    - Match v to u with probability $\alpha_u(v) = \frac{1}{2-\sum_{r<v} y_{(r,u)}}$



Algorithm for Vertex Arrivals

- Precompute: $y_e = \Pr[e \in OPT]$
- Upon arrival of vertex v:
  - B_v ≔ edges from v to former vertices; $-B_v \coloneqq E \setminus \mathrm{B}_v$
  - Observe weights w_e of new edges B_v
  - Sample weights $\tilde{w}_{e'}$ for edges $e' \in -B_v$
  - u ≔ partner of v in max-weight matching on $\left(w_{B_v}, \tilde{w}_{-B_v}\right)$
  - If (a) u exists, (b) $u < v$, (c) u is available, then
    - Match v to u with probability $\alpha_u(v) = \frac{1}{2 - \sum_{r<v} y_{(r,u)}}$

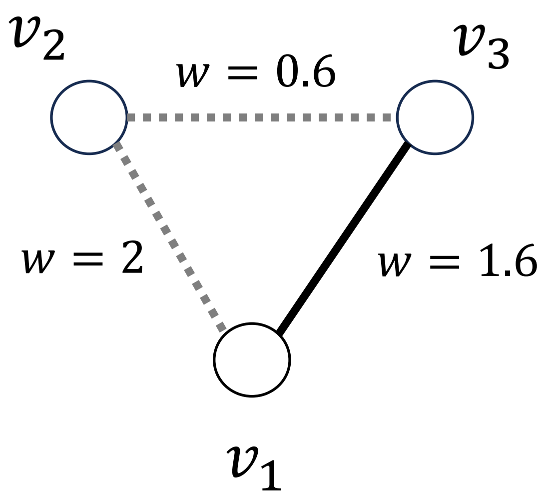


ALG = 1.6
OPT = 2

Algorithm for Vertex Arrivals

- Precompute: $y_e = \Pr[e \in OPT]$
- Upon arrival of vertex v:
  - $B_v :=$ edges from v to former vertices; $-B_v := E \backslash B_v$
  - Observe weights w_e of new edges B_v
  - Sample weights $\widetilde{w}_{e'}$ for edges $e' \in -B_v$
  - $u :=$ partner of v in max-weight matching on $\left(w_{B_v}, \widetilde{w}_{-B_v}\right)$
  - If (a) u exists, (b) $u < v$, (c) u is available, then
    - Match v to u with probability $\alpha_u(v) = \frac{1}{2 - \sum_{r<v} y_{(r,u)}}$

Note: $\alpha_u(v) \leq 1$ because $\sum_{r<v} y_{(r,u)} \leq 1$

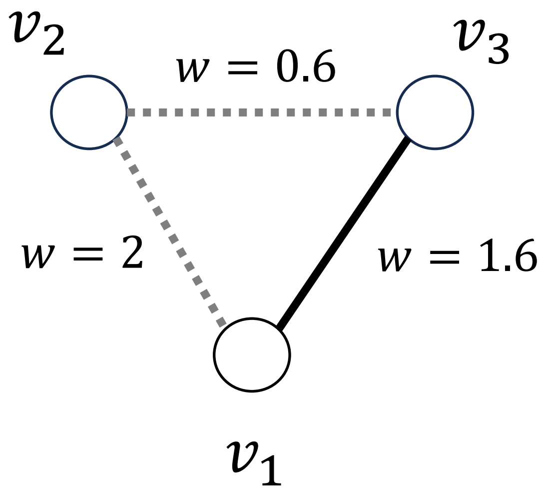


ALG = 1.6
OPT = 2

Algorithm for Vertex Arrivals

- Precompute: $y_e = \Pr[e \in OPT]$
- Upon arrival of vertex v:
  - $B_v :=$ edges from v to former vertices; $-B_v := E \setminus B_v$
  - Observe weights w_e of new edges B_v
  - Sample weights $\widetilde{w}_{e'}$ for edges $e' \in -B_v$
  - $u :=$ partner of v in max-weight matching on $(w_{B_v}, \widetilde{w}_{-B_v})$
  - If (a) u exists, (b) $u < v$, ~~(c) u is available~~, then
    - Match v to u ~~with probability~~ $\cancel{\alpha_u(v) = \frac{1}{2-\sum_{r<v} y_{(r,u)}}}$

"provisional edges"



ALG = 1.6
OPT = 2

Proof Outline

Lemma 1. $\Pr[e \text{ is provisional}] = \Pr[e \in OPT] = y_e$

Proof Outline

Lemma 1. $\Pr[e \text{ is provisional}] = \Pr[e \in OPT] = y_e$

Lemma 2. $\Pr[e \text{ is matched} \mid e \text{ is provisional}] = \frac{1}{2}$

Proof Outline

Lemma 1. $\Pr[e \text{ is provisional}] = \Pr[e \in OPT] = y_e$

Lemma 2. $\Pr[e \text{ is matched} \mid e \text{ is provisional}] = \frac{1}{2}$

Proof: By induction:

$$\underbrace{\left(1 - \frac{\sum_{r<v} y_{(r,u)}}{2}\right)}_{u \text{ is available}} \cdot \underbrace{\left(\frac{1}{2 - \sum_{r<v} y_{(r,u)}}\right)}_{\alpha_u(v)} = \frac{1}{2}$$

Q.E.D.

Proof Outline

Lemma 1. $\Pr[e \text{ is provisional}] = \Pr[e \in OPT] = y_e$

Lemma 2. $\Pr[e \text{ is matched} \mid e \text{ is provisional}] = \frac{1}{2}$

Lemma 3. Expected value of provisional edges is $\mathbb{E}[OPT]$

Proof Outline

Lemma 1. $\Pr[e \text{ is provisional}] = \Pr[e \in OPT] = y_e$

Lemma 2. $\Pr[e \text{ is matched} \mid e \text{ is provisional}] = \frac{1}{2}$

Lemma 3. Expected value of provisional edges is $\mathbb{E}[OPT]$

Conclusion: ALG has a competitive ratio of $1/2$.

Q.E.D.

Additional Directions

- Better understanding of random-order OCRS for matching and other problems. Recent progress in [MacRury Ma 2024], but not yet fully understood.
- Infinite-time horizon prophet inequalities (cf. the Stationary Prophet Inequality Problem). Has connections to (offline) contention resolution schemes (CRS) [Kessel Shameli Patel Saberi Wajc '22, Patel Wajc '24]
- Online correlated selection (OCS) as in [Fahrbach, Huang, Tao, and Zadimoghaddam '20] (and follow-up). Has some connection to OCRS. Making this connection tighter and more explicit is an interesting direction.
- Online dependent rounding. One can do better than offline single-item CRS (1.519), but no better than $1/(2\sqrt{2} - 2) \approx 1.208$. What's the right answer? Known techniques relate to philosopher inequality, and online edge coloring multi-graphs. [Naor Srinivasan Wajc 23+]

Summary

- Alternative proof for single-choice prophet inequality
  - via: “online contention resolution”
- Prophet inequalities for online matching via this technique
  - with edge arrivals in general graphs
  - with vertex (“batched”) arrivals in general graphs

Thanks! Coffee!