Online Bipartite Matching

Calum MacRury

Department of Computer Science, University of Toronto, Toronto ON, Canada

Adversarial Arrivals

Adversarial Arrivals

- Input: a **bipartite graph** $G = (U, V, E)$ with n online nodes V.

Adversarial Arrivals

- Input: a **bipartite graph** $G = (U, V, E)$ with n online nodes V.
- Online algorithm $\mathcal{A}$ given only U initially.

Adversarial Arrivals

- Input: a **bipartite graph** $G = (U, V, E)$ with n online nodes V.
- Online algorithm $\mathcal{A}$ given only U initially.
- Vertices V arrive based on **adversarial ordering** π.

Adversarial Arrivals

- Input: a **bipartite graph** $G = (U, V, E)$ with n online nodes V.
- Online algorithm $\mathcal{A}$ given only U initially.
- Vertices V arrive based on **adversarial ordering** π. Upon arrival of vertex $v \in V$, the neighbourhood N_v of v is revealed to $\mathcal{A}$.

Adversarial Arrivals

- Input: a **bipartite graph** $G = (U, V, E)$ with n online nodes V.
- Online algorithm $\mathcal{A}$ given only U initially.
- Vertices V arrive based on **adversarial ordering** π. Upon arrival of vertex $v \in V$, the neighbourhood N_v of v is revealed to $\mathcal{A}$.
- The algorithm makes an **irrevocable decision** as to whether or not to match v, based on all currently available information.

Adversarial Arrivals

- **Input:** a **bipartite graph** $G = (U, V, E)$ with n online nodes V.
- Online algorithm $\mathcal{A}$ given only U initially.
- Vertices V arrive based on **adversarial ordering** π. Upon arrival of vertex $v \in V$, the neighbourhood N_v of v is revealed to $\mathcal{A}$.
- The algorithm makes an **irrevocable decision** as to whether or not to match v, based on all currently available information.
- This decision can be made either deterministically, or using randomization.

Adversarial Arrivals

- **Input:** a **bipartite graph** $G = (U, V, E)$ with n online nodes V.
- Online algorithm $\mathcal{A}$ given only U initially.
- Vertices V arrive based on **adversarial ordering** π. Upon arrival of vertex $v \in V$, the neighbourhood N_v of v is revealed to $\mathcal{A}$.
- The algorithm makes an **irrevocable decision** as to whether or not to match v, based on all currently available information.
- This decision can be made either deterministically, or using randomization.
- **Output:** a matching $\mathcal{A}(G, \pi)$ of G.

Adversarial Arrivals

- **Input:** a **bipartite graph** $G = (U, V, E)$ with n online nodes V.
- Online algorithm $\mathcal{A}$ given only U initially.
- Vertices V arrive based on **adversarial ordering** π. Upon arrival of vertex $v \in V$, the neighbourhood N_v of v is revealed to $\mathcal{A}$.
- The algorithm makes an **irrevocable decision** as to whether or not to match v, based on all currently available information.
- This decision can be made either deterministically, or using randomization.
- **Output:** a matching $\mathcal{A}(G, \pi)$ of G.
- **Goal:** maximize $|\mathcal{A}(G, \pi)|$, or $\mathbb{E}[|\mathcal{A}(G, \pi)|]$.

Benchmarking via Competitive Ratios

- If $\mathcal{A}$ is deterministic then the **competitive ratio** of the algorithm is defined as

$$\inf_{G,\pi} \frac{|\mathcal{A}(G,\pi)|}{\mathrm{OPT}(G)},$$

where $\mathrm{OPT}(G)$ is the size of a maximum matching of G.

Benchmarking via Competitive Ratios

- If $\mathcal{A}$ is deterministic then the **competitive ratio** of the algorithm is defined as

$$\inf_{G,\pi} \frac{|\mathcal{A}(G,\pi)|}{\mathrm{OPT}(G)},$$

  where $\mathrm{OPT}(G)$ is the size of a maximum matching of G.
- If $\mathcal{A}$ is randomized, then the competitive ratio is

$$\inf_{G,\pi} \frac{\mathbb{E}[|\mathcal{A}(G,\pi)|]}{\mathrm{OPT}(G)}.$$

Benchmarking via Competitive Ratios

- If $\mathcal{A}$ is deterministic then the **competitive ratio** of the algorithm is defined as

$$\inf_{G,\pi} \frac{|\mathcal{A}(G,\pi)|}{\mathrm{OPT}(G)},$$

  where $\mathrm{OPT}(G)$ is the size of a maximum matching of G.

- If $\mathcal{A}$ is randomized, then the competitive ratio is

$$\inf_{G,\pi} \frac{\mathbb{E}[|\mathcal{A}(G,\pi)|]}{\mathrm{OPT}(G)}.$$

- The primary goal of online algorithms is to attain competitive ratios as large as possible.

Deterministic Algorithms

- **Greedy** initially specifies an ordering $\lambda : U \to \{1, \ldots, |U|\}$ on its offline nodes.

Deterministic Algorithms

- **Greedy** initially specifies an ordering $\lambda : U \to \{1, \ldots, |U|\}$ on its offline nodes.
- When an online node v arrives, **Greedy** then attempts to match v to the available vertex $u \in N_v$ for which $\lambda(u)$ is minimal.

Deterministic Algorithms

- **Greedy** initially specifies an ordering $\lambda : U \to \{1, \ldots, |U|\}$ on its offline nodes.
- When an online node v arrives, **Greedy** then attempts to match v to the available vertex $u \in N_v$ for which $\lambda(u)$ is minimal.
- Any choice of λ yields an algorithm with competitive ratio 1/2.

Deterministic Algorithms

- **Greedy** initially specifies an ordering $\lambda : U \to \{1, \ldots, |U|\}$ on its offline nodes.
- When an online node v arrives, **Greedy** then attempts to match v to the available vertex $u \in N_v$ for which $\lambda(u)$ is minimal.
- Any choice of λ yields an algorithm with competitive ratio 1/2.
- This is provably best amongst all *deterministic* online algorithms.

Improvement Through Randomization

- One can improve on 1/2 via randomization.

Improvement Through Randomization

- One can improve on 1/2 via randomization.
- **Ranking** draws λ uniformly at random (u.a.r.), and then executes **Greedy** with the ordering λ.

Improvement Through Randomization

- One can improve on 1/2 via randomization.
- **Ranking** draws λ uniformly at random (u.a.r.), and then executes **Greedy** with the ordering λ.

Karp, Vazirani, and Vazirani (1990)

Ranking attains a competitive ratio of $1 - 1/e$.

Improvement Through Randomization

- One can improve on 1/2 via randomization.
- **Ranking** draws λ uniformly at random (u.a.r.), and then executes **Greedy** with the ordering λ.

Karp, Vazirani, and Vazirani (1990)

Ranking attains a competitive ratio of $1 - 1/e$.

Karp, Vazirani, and Vazirani (1990)

$1 - 1/e$ is optimal amongst all online algorithms.

Generalizing to Offline Vertex Weights

- Suppose that G has offline vertex weights $(w_u)_{u \in U}$.

Generalizing to Offline Vertex Weights

- Suppose that G has offline vertex weights $(w_u)_{u \in U}$.
- Goal of the online algorithm is to build a matching $\mathcal{M}$ whose weight $w(\mathcal{M}) := \sum_{e=(u,v) \in \mathcal{M}} w_u$ is maximized.

Generalizing to Offline Vertex Weights

- Suppose that G has offline vertex weights $(w_u)_{u \in U}$.
- Goal of the online algorithm is to build a matching $\mathcal{M}$ whose weight $w(\mathcal{M}) := \sum_{e=(u,v)\in\mathcal{M}} w_u$ is maximized.
- Algorithm is now benchmarked against the maximum weight of a matching of G, again denoted by $\mathrm{OPT}(G)$.

Generalizing to Offline Vertex Weights

- Suppose that G has offline vertex weights $(w_u)_{u \in U}$.
- Goal of the online algorithm is to build a matching $\mathcal{M}$ whose weight $w(\mathcal{M}) := \sum_{e=(u,v)\in\mathcal{M}} w_u$ is maximized.
- Algorithm is now benchmarked against the maximum weight of a matching of G, again denoted by $\mathrm{OPT}(G)$.
- We know $1 - 1/e$ is the best possible competitive ratio.

A Pricing Based Interpretation of Ranking

- Consider the algorithm as a **seller** of the items U.

A Pricing Based Interpretation of Ranking

- Consider the algorithm as a **seller** of the items U.
- View V as a collection of **buyers**, where $v \in V$ is interested in items N_v, and values them according to the weights $(w_u)_{u \in N_v}$.

A Pricing Based Interpretation of Ranking

- Consider the algorithm as a **seller** of the items U.
- View V as a collection of **buyers**, where $v \in V$ is interested in items N_v, and values them according to the weights $(w_u)_{u \in N_v}$.
- Fix an increasing function $g : [0,1] \to [0,1]$ with $g(1) = 1$.

A Pricing Based Interpretation of Ranking

- Consider the algorithm as a **seller** of the items U.
- View V as a collection of **buyers**, where $v \in V$ is interested in items N_v, and values them according to the weights $(w_u)_{u \in N_v}$.
- Fix an increasing function $g : [0,1] \to [0,1]$ with $g(1) = 1$.
- For each $u \in U$, draw **rank** $X_u \sim \mathcal{U}[0,1]$ independently.

A Pricing Based Interpretation of Ranking

- Consider the algorithm as a **seller** of the items U.
- View V as a collection of **buyers**, where $v \in V$ is interested in items N_v, and values them according to the weights $(w_u)_{u \in N_v}$.
- Fix an increasing function $g : [0,1] \to [0,1]$ with $g(1) = 1$.
- For each $u \in U$, draw **rank** $X_u \sim \mathcal{U}[0,1]$ independently.
- Set $p_u := w_u \cdot g(X_u)$ to be the price of item u.

A Pricing Based Interpretation of Ranking

- Consider the algorithm as a **seller** of the items U.
- View V as a collection of **buyers**, where $v \in V$ is interested in items N_v, and values them according to the weights $(w_u)_{u \in N_v}$.
- Fix an increasing function $g : [0,1] \to [0,1]$ with $g(1) = 1$.
- For each $u \in U$, draw **rank** $X_u \sim \mathcal{U}[0,1]$ independently.
- Set $p_u := w_u \cdot g(X_u)$ to be the price of item u.
- Upon arrival of $v \in V$, buyer v purchases the item $u \in N_v$ such that $w_u - p_u = w_u \cdot (1 - g(X_u))$ is maximized.

A Pricing Based Interpretation of Ranking

- Consider the algorithm as a **seller** of the items U.
- View V as a collection of **buyers**, where $v \in V$ is interested in items N_v, and values them according to the weights $(w_u)_{u \in N_v}$.
- Fix an increasing function $g : [0,1] \to [0,1]$ with $g(1) = 1$.
- For each $u \in U$, draw **rank** $X_u \sim \mathcal{U}[0,1]$ independently.
- Set $p_u := w_u \cdot g(X_u)$ to be the price of item u.
- Upon arrival of $v \in V$, buyer v purchases the item $u \in N_v$ such that $w_u - p_u = w_u \cdot (1 - g(X_u))$ is maximized. Let λ be the induced ordering.

A Pricing Based Interpretation of Ranking

- Consider the algorithm as a **seller** of the items U.
- View V as a collection of **buyers**, where $v \in V$ is interested in items N_v, and values them according to the weights $(w_u)_{u \in N_v}$.
- Fix an increasing function $g : [0,1] \to [0,1]$ with $g(1) = 1$.
- For each $u \in U$, draw **rank** $X_u \sim \mathcal{U}[0,1]$ independently.
- Set $p_u := w_u \cdot g(X_u)$ to be the price of item u.
- Upon arrival of $v \in V$, buyer v purchases the item $u \in N_v$ such that $w_u - p_u = w_u \cdot (1 - g(X_u))$ is maximized. Let λ be the induced ordering.
- Observe λ is u.a.r. if the weights are identical.

A Pricing Based Interpretation of Ranking

- Consider the algorithm as a **seller** of the items U.
- View V as a collection of **buyers**, where $v \in V$ is interested in items N_v, and values them according to the weights $(w_u)_{u \in N_v}$.
- Fix an increasing function $g : [0, 1] \to [0, 1]$ with $g(1) = 1$.
- For each $u \in U$, draw **rank** $X_u \sim \mathcal{U}[0, 1]$ independently.
- Set $p_u := w_u \cdot g(X_u)$ to be the price of item u.
- Upon arrival of $v \in V$, buyer v purchases the item $u \in N_v$ such that $w_u - p_u = w_u \cdot (1 - g(X_u))$ is maximized. Let λ be the induced ordering.
- Observe λ is u.a.r. if the weights are identical.
- Let $\mathcal{M}$ be the matching returned. Define Revenue $= \sum_{e=(u,v) \in \mathcal{M}} p_u$ and Utility $= \sum_{e=(u,v) \in \mathcal{M}} (w_u - p_u)$.

A Pricing Based Interpretation of Ranking

- Consider the algorithm as a **seller** of the items U.
- View V as a collection of **buyers**, where $v \in V$ is interested in items N_v, and values them according to the weights $(w_u)_{u \in N_v}$.
- Fix an increasing function $g : [0,1] \to [0,1]$ with $g(1) = 1$.
- For each $u \in U$, draw **rank** $X_u \sim \mathcal{U}[0,1]$ independently.
- Set $p_u := w_u \cdot g(X_u)$ to be the price of item u.
- Upon arrival of $v \in V$, buyer v purchases the item $u \in N_v$ such that $w_u - p_u = w_u \cdot (1 - g(X_u))$ is maximized. Let λ be the induced ordering.
- Observe λ is u.a.r. if the weights are identical.
- Let $\mathcal{M}$ be the matching returned. Define Revenue $= \sum_{e=(u,v) \in \mathcal{M}} p_u$ and Utility $= \sum_{e=(u,v) \in \mathcal{M}} (w_u - p_u)$.
- Observe that $w(\mathcal{M})$ measures the **social welfare** (overall good) of matching items U to V via $\mathcal{M}$.

Require: U with offline vertex weights $w = (w_u)_{u \in U}$.

Ensure: a matching $\mathcal{M}$ of (unknown) vertex weighted graph $G = (U, V, E)$.

1: Independently draw $X_u \sim \mathcal{U}[0, 1]$ for each $u \in U$.

2: Compute an ordering λ which ranks $u \in U$ in decreasing order of $w_u(1 - g(X_u))$, where $g(x) := \exp(x - 1)$

Require: U with offline vertex weights $w = (w_u)_{u \in U}$.

Ensure: a matching $\mathcal{M}$ of (unknown) vertex weighted graph $G = (U, V, E)$.

1: Independently draw $X_u \sim \mathcal{U}[0, 1]$ for each $u \in U$.

2: Compute an ordering λ which ranks $u \in U$ in decreasing order of $w_u(1 - g(X_u))$, where $g(x) := \exp(x - 1)$

3: $\mathcal{M} \leftarrow \emptyset$.

4: $R \leftarrow U$. ▷ remaining vertices.

The Weighted-Ranking Algorithm

Require: U with offline vertex weights $w = (w_u)_{u \in U}$.
Ensure: a matching $\mathcal{M}$ of (unknown) vertex weighted graph $G = (U, V, E)$.

Independently draw $X_u \sim \mathcal{U}[0,1]$ for each $u \in U$.
Compute an ordering λ which ranks $u \in U$ in decreasing order of $w_u(1 - g(X_u))$, where $g(x) := \exp(x-1)$
$\mathcal{M} \leftarrow \emptyset$.
$R \leftarrow U$. ▷ remaining vertices.
for $t = 1, \dots, n$ **do**
    Let v_t be the current online arrival.
    if $N_{v_t} \cap R \neq \emptyset$ **then**
        Set $\mathcal{M}(v_t) = u$, where $\lambda(u)$ is the smallest integer amongst $N_{v_t} \cap R$
        $R \leftarrow R \setminus u$.
    end if
end for
Return $\mathcal{M}$.

The Weighted-Ranking Algorithm

- The choice of $g(x) := \exp(x-1)$ is due to Aggarwal et al. 2011.

- The choice of $g(x) := \exp(x - 1)$ is due to Aggarwal et al. 2011.

Aggarwal et al. 2011

Weighted-Ranking attains a competitive ratio of $1 - 1/e$.

The Weighted-Ranking Algorithm

- The choice of $g(x) := \exp(x - 1)$ is due to Aggarwal et al. 2011.

Aggarwal et al. 2011

Weighted-Ranking attains a competitive ratio of $1 - 1/e$.

- Devanur et al. (2013) provide an (alternative) primal-dual analysis which leverages the pricing based interpretation to greatly simplify the analysis of Aggarwal et al.

The Random Order Model

- Since $1 - 1/e$ is the optimal competitive ratio, many works have studied more optimistic online matching models in order to surpass this barrier.

- Since $1 - 1/e$ is the optimal competitive ratio, many works have studied more optimistic online matching models in order to surpass this barrier.
- In the **Random Order Model (ROM)**, the nodes V of G are presented to $\mathcal{A}$ uniformly at random, opposed to in an adversarial order, and so the matching $\mathcal{A}(G)$ returned by $\mathcal{A}$ is random.

The Random Order Model

- Since $1 - 1/e$ is the optimal competitive ratio, many works have studied more optimistic online matching models in order to surpass this barrier.
- In the **Random Order Model (ROM)**, the nodes V of G are presented to $\mathcal{A}$ uniformly at random, opposed to in an adversarial order, and so the matching $\mathcal{A}(G)$ returned by $\mathcal{A}$ is random.
- Performance of $\mathcal{A}$, namely $\mathbb{E}[w(\mathcal{A}(G))]$, is averaged over π.

The Random Order Model

- Since $1 - 1/e$ is the optimal competitive ratio, many works have studied more optimistic online matching models in order to surpass this barrier.
- In the **Random Order Model (ROM)**, the nodes V of G are presented to $\mathcal{A}$ uniformly at random, opposed to in an adversarial order, and so the matching $\mathcal{A}(G)$ returned by $\mathcal{A}$ is random.
- Performance of $\mathcal{A}$, namely $\mathbb{E}[w(\mathcal{A}(G))]$, is averaged over π.
- The competitive ratio of $\mathcal{A}$ in the random order model is then

$$\inf_G \frac{\mathbb{E}[w(\mathcal{A}(G))]}{\mathrm{OPT}(G)}.$$

The Random Order Model

- Since $1 - 1/e$ is the optimal competitive ratio, many works have studied more optimistic online matching models in order to surpass this barrier.
- In the **Random Order Model (ROM)**, the nodes V of G are presented to $\mathcal{A}$ uniformly at random, opposed to in an adversarial order, and so the matching $\mathcal{A}(G)$ returned by $\mathcal{A}$ is random.
- Performance of $\mathcal{A}$, namely $\mathbb{E}[w(\mathcal{A}(G))]$, is averaged over π.
- The competitive ratio of $\mathcal{A}$ in the random order model is then

$$\inf_G \frac{\mathbb{E}[w(\mathcal{A}(G))]}{\mathrm{OPT}(G)}.$$

- Observe its competitive ratio is no smaller than in the adversarial order model.

The Random Order Model

- Since $1 - 1/e$ is the optimal competitive ratio, many works have studied more optimistic online matching models in order to surpass this barrier.
- In the **Random Order Model (ROM)**, the nodes V of G are presented to $\mathcal{A}$ uniformly at random, opposed to in an adversarial order, and so the matching $\mathcal{A}(G)$ returned by $\mathcal{A}$ is random.
- Performance of $\mathcal{A}$, namely $\mathbb{E}[w(\mathcal{A}(G))]$, is averaged over π.
- The competitive ratio of $\mathcal{A}$ in the random order model is then

$$\inf_G \frac{\mathbb{E}[w(\mathcal{A}(G))]}{\mathrm{OPT}(G)}.$$

- Observe its competitive ratio is no smaller than in the adversarial order model.
- A common technique in the literature is to view the vertices of V as arriving in increasing order of $(Y_v)_{v \in V}$, where $Y_v \sim \mathcal{U}[0,1]$ is the **arrival time** of v.

Randomized Algorithms in ROM: Unweighted

Mahdian et al. 2011

Ranking achieves a competitive ratio of 0.696 in the unweighted ROM setting.

Randomized Algorithms in ROM: Unweighted

Mahdian et al. 2011

Ranking achieves a competitive ratio of 0.696 in the unweighted ROM setting.

- Proof utilizes strongly factor revealing linear programs (LP)s.

Randomized Algorithms in ROM: Unweighted

Mahdian et al. 2011

Ranking achieves a competitive ratio of 0.696 in the unweighted ROM setting.

- Proof utilizes strongly factor revealing linear programs (LP)s.
- Techniques don't seem to extend to the vertex-weighted setting.

- Huang et al. 2018 introduce a generalization of Weighted-Ranking, defined using a function $g(x, y)$, where $g : [0, 1]^2 \rightarrow [0, 1]$.

- Huang et al. 2018 introduce a generalization of Weighted-Ranking, defined using a function $g(x, y)$, where $g : [0, 1]^2 \to [0, 1]$.
- The price of $u \in U$ is then $w_u \cdot g(X_u, Y_v)$ where $X_u \in \mathcal{U}[0, 1]$ is the rank of $u \in U$, and $Y_v \in \mathcal{U}[0, 1]$ is the arrival time of vertex $v \in V$.

- Huang et al. 2018 introduce a generalization of Weighted-Ranking, defined using a function $g(x, y)$, where $g : [0, 1]^2 \to [0, 1]$.
- The price of $u \in U$ is then $w_u \cdot g(X_u, Y_v)$ where $X_u \in \mathcal{U}[0, 1]$ is the rank of $u \in U$, and $Y_v \in \mathcal{U}[0, 1]$ is the arrival time of vertex $v \in V$.
- For fixed $x \in [0, 1]$, if $y_1 < y_2$, then $g(x, y_2) < g(x, y_1)$.

Randomized Algorithms in ROM: Vertex Weighted

- Huang et al. 2018 introduce a generalization of Weighted-Ranking, defined using a function $g(x, y)$, where $g : [0, 1]^2 \to [0, 1]$.
- The price of $u \in U$ is then $w_u \cdot g(X_u, Y_v)$ where $X_u \in \mathcal{U}[0, 1]$ is the rank of $u \in U$, and $Y_v \in \mathcal{U}[0, 1]$ is the arrival time of vertex $v \in V$.
- For fixed $x \in [0, 1]$, if $y_1 < y_2$, then $g(x, y_2) < g(x, y_1)$. Thus, the later a vertex v arrives (larger Y_v is), the lower the prices for v.

Randomized Algorithms in ROM: Vertex Weighted

- Huang et al. 2018 introduce a generalization of Weighted-Ranking, defined using a function $g(x, y)$, where $g : [0, 1]^2 \to [0, 1]$.
- The price of $u \in U$ is then $w_u \cdot g(X_u, Y_v)$ where $X_u \in \mathcal{U}[0, 1]$ is the rank of $u \in U$, and $Y_v \in \mathcal{U}[0, 1]$ is the arrival time of vertex $v \in V$.
- For fixed $x \in [0, 1]$, if $y_1 < y_2$, then $g(x, y_2) < g(x, y_1)$. Thus, the later a vertex v arrives (larger Y_v is), the lower the prices for v.
- This property balances the fact that later buyers naturally have fewer options available.

Randomized Algorithms in ROM: Vertex Weighted

- Huang et al. 2018 introduce a generalization of Weighted-Ranking, defined using a function $g(x, y)$, where $g : [0, 1]^2 \to [0, 1]$.
- The price of $u \in U$ is then $w_u \cdot g(X_u, Y_v)$ where $X_u \in \mathcal{U}[0, 1]$ is the rank of $u \in U$, and $Y_v \in \mathcal{U}[0, 1]$ is the arrival time of vertex $v \in V$.
- For fixed $x \in [0, 1]$, if $y_1 < y_2$, then $g(x, y_2) < g(x, y_1)$. Thus, the later a vertex v arrives (larger Y_v is), the lower the prices for v.
- This property balances the fact that later buyers naturally have fewer options available.

Huang et al. 2018

Generalized-Ranking achieves a competitive ratio of 0.6534 in the vertex-weighted ROM setting.

Open Problems

- What is the best competitive ratio attainable in the unweighted ROM setting? Is the Ranking algorithm optimal?

Open Problems

- What is the best competitive ratio attainable in the unweighted ROM setting? Is the Ranking algorithm optimal?
- In the vertex weighted setting, **0.654** was recently improved upon by Jin and Williamson (2020) to **0.6629** via a different pricing function $g(x, y)$.

Open Problems

- What is the best competitive ratio attainable in the unweighted ROM setting? Is the Ranking algorithm optimal?
- In the vertex weighted setting, **0.654** was recently improved upon by Jin and Williamson (2020) to **0.6629** via a different pricing function $g(x, y)$. What is the optimal competitive ratio attainable via algorithms of this form?

Open Problems

- What is the best competitive ratio attainable in the unweighted ROM setting? Is the Ranking algorithm optimal?
- In the vertex weighted setting, 0.654 was recently improved upon by Jin and Williamson (2020) to 0.6629 via a different pricing function $g(x, y)$. What is the optimal competitive ratio attainable via algorithms of this form?
- 0.823 is the best known upper bound (negative result) even in the unweighted setting.

Open Problems

- What is the best competitive ratio attainable in the unweighted ROM setting? Is the Ranking algorithm optimal?
- In the vertex weighted setting, **0.654** was recently improved upon by Jin and Williamson (2020) to **0.6629** via a different pricing function $g(x, y)$. What is the optimal competitive ratio attainable via algorithms of this form?
- **0.823** is the best known upper bound (negative result) even in the unweighted setting. Can this be improved substantially?

Online Matching with Edge Weights

- We have so far only considered the case when $G = (U, V, E)$ is vertex weighted.

Online Matching with Edge Weights

- We have so far only considered the case when $G = (U, V, E)$ is vertex weighted.
- All the arrival models and corresponding competitive ratios generalize to the edge weighted setting. I.e., G has edge weights $(w_e)_{e \in E}$.

Online Matching with Edge Weights

- We have so far only considered the case when $G = (U, V, E)$ is vertex weighted.
- All the arrival models and corresponding competitive ratios generalize to the edge weighted setting. I.e., G has edge weights $(w_e)_{e \in E}$.
- However, in the adversarial arrival model, no algorithm attains a constant competitive ratio.

Online Matching with Edge Weights

- We have so far only considered the case when $G = (U, V, E)$ is vertex weighted.
- All the arrival models and corresponding competitive ratios generalize to the edge weighted setting. I.e., G has edge weights $(w_e)_{e \in E}$.
- However, in the adversarial arrival model, no algorithm attains a constant competitive ratio.
- This is true even when $|U| = 1$ and $|V| = 2$.

Online Matching with Edge Weights

- We have so far only considered the case when $G = (U, V, E)$ is vertex weighted.
- All the arrival models and corresponding competitive ratios generalize to the edge weighted setting. I.e., G has edge weights $(w_e)_{e \in E}$.
- However, in the adversarial arrival model, no algorithm attains a constant competitive ratio.
- This is true even when $|U| = 1$ and $|V| = 2$. Consider when $w_{e_1} \ll w_{e_2}$.

Online Matching with Edge Weights

- We have so far only considered the case when $G = (U, V, E)$ is vertex weighted.
- All the arrival models and corresponding competitive ratios generalize to the edge weighted setting. I.e., G has edge weights $(w_e)_{e \in E}$.
- However, in the adversarial arrival model, no algorithm attains a constant competitive ratio.
- This is true even when $|U| = 1$ and $|V| = 2$. Consider when $w_{e_1} \ll w_{e_2}$.
- In the ROM setting, constant competitive ratios *can* be attained.

- When $|U| = 1$, and $E = \{u\} \times V$, observe that $\mathrm{OPT}(G) = \max_{e \in E} w_e$.

The Secretary Problem

- When $|U| = 1$, and $E = \{u\} \times V$, observe that $\mathrm{OPT}(G) = \max_{e \in E} w_e$.
- Moreover, the algorithm can select at most one edge, and the edges arrive uniformly at random.

- When $|U| = 1$, and $E = \{u\} \times V$, observe that $\mathrm{OPT}(G) = \max_{e \in E} w_e$.
- Moreover, the algorithm can select at most one edge, and the edges arrive uniformly at random.
- The **Secretary** algorithm is simple:

- When $|U| = 1$, and $E = \{u\} \times V$, observe that $\text{OPT}(G) = \max_{e \in E} w_e$.
- Moreover, the algorithm can select at most one edge, and the edges arrive uniformly at random.
- The **Secretary** algorithm is simple:
- Pass on the first n/e arriving edges (where $n = |V|$).

The Secretary Problem

- When $|U| = 1$, and $E = \{u\} \times V$, observe that $\mathrm{OPT}(G) = \max_{e \in E} w_e$.
- Moreover, the algorithm can select at most one edge, and the edges arrive uniformly at random.
- The **Secretary** algorithm is simple:
- Pass on the first n/e arriving edges (where $n = |V|$).
- Afterwards, accept the first edge whose weight is at least as large as the first n/e edges.

- When $|U| = 1$, and $E = \{u\} \times V$, observe that $\mathrm{OPT}(G) = \max_{e \in E} w_e$.
- Moreover, the algorithm can select at most one edge, and the edges arrive uniformly at random.
- The **Secretary** algorithm is simple:
- Pass on the first n/e arriving edges (where $n = |V|$).
- Afterwards, accept the first edge whose weight is at least as large as the first n/e edges.

Gardner (1960), Dynkin (1963)

Secretary attains an asymptotic (as $n \to \infty$) competitive ratio of $1/e$ for random order arrivals, and this is best possible.

The Secretary Problem

- When $|U| = 1$, and $E = \{u\} \times V$, observe that $\mathrm{OPT}(G) = \max_{e \in E} w_e$.
- Moreover, the algorithm can select at most one edge, and the edges arrive uniformly at random.
- The **Secretary** algorithm is simple:
- Pass on the first n/e arriving edges (where $n = |V|$).
- Afterwards, accept the first edge whose weight is at least as large as the first n/e edges.

Gardner (1960), Dynkin (1963)

Secretary attains an asymptotic (as $n \to \infty$) competitive ratio of $1/e$ for random order arrivals, and this is best possible.

- Analysis of **Secretary** is fairly immediate. Hardness result is more involved.

- Since $1/e$ is best possible, various works have focused on generalizing the problem to more sophisticated online settings.

- Since $1/e$ is best possible, various works have focused on generalizing the problem to more sophisticated online settings.
- There is a natural modification of the secretary algorithm to the matching setting called the Secretary-Matching algorithm.

- Since $1/e$ is best possible, various works have focused on generalizing the problem to more sophisticated online settings.
- There is a natural modification of the secretary algorithm to the matching setting called the **Secretary-Matching** algorithm.

Kesselheim et al. (2013)

Secretary-Matching attains an asymptotic (as $|V| \to \infty$) competitive ratio of $1/e$.

Require: U and $n := |V|$.

Ensure: a matching $\mathcal{M}$ from (unknown) edge weighted graph $G = (U, V, E)$.

1: Set $\mathcal{M} \leftarrow \emptyset$.

2: Set $G_0 = (U, \emptyset, \emptyset)$

Secretary Matching Algorithm

Require: U and $n := |V|$.

Ensure: a matching $\mathcal{M}$ from (unknown) edge weighted graph $G = (U, V, E)$.

Set $\mathcal{M} \leftarrow \emptyset$.
Set $G_0 = (U, \emptyset, \emptyset)$
for $t = 1, \ldots, n$ **do**
    Input v_t, and compute G_t by updating G_{t-1} to contain v_t.
    if $t < \lfloor n/e \rfloor$ **then**
        Pass on v_t.

Secretary Matching Algorithm

Require: U and $n := |V|$.

Ensure: a matching $\mathcal{M}$ from (unknown) edge weighted graph $G = (U, V, E)$.

Set $\mathcal{M} \leftarrow \emptyset$.
Set $G_0 = (U, \emptyset, \emptyset)$
for $t = 1, \ldots, n$ **do**
    Input v_t, and compute G_t by updating G_{t-1} to contain v_t.
    if $t < \lfloor n/e \rfloor$ **then**
        Pass on v_t.
    else
        Compute an optimal matching $\mathcal{M}_t$ of G_t
        Set e_t to be the edge matched to v_t via $\mathcal{M}_t$.
        if $e_t = (u_t, v_t)$ exists and u_t is unmatched **then**
            Add e_t to $\mathcal{M}$.
        end if
    end if
end for
return $\mathcal{M}$.

Online Stochastic Matching

- Since $1/e$ is best possible, various works have focused on understanding even more optimistic online models than ROM.

Online Stochastic Matching

- Since **1/e** is best possible, various works have focused on understanding even more optimistic online models than ROM.
- A popular setting is the **prophet matching problem with known i.i.d. arrivals** which considers when $G = (U, V, E)$ is drawn from a distribution.

Online Stochastic Matching

- Since **1/e** is best possible, various works have focused on understanding even more optimistic online models than ROM.
- A popular setting is the **prophet matching problem with known i.i.d. arrivals** which considers when $G = (U, V, E)$ is drawn from a distribution.
- The adversary now only gets to select a **type graph** $H_{\text{typ}} = (U, B, F)$, a **known** distribution $\mathcal{D}$ supported on B, and the number of arrivals $n \geq 1$.

Online Stochastic Matching

- Since **1/e** is best possible, various works have focused on understanding even more optimistic online models than ROM.
- A popular setting is the **prophet matching problem with known i.i.d. arrivals** which considers when $G = (U, V, E)$ is drawn from a distribution.
- The adversary now only gets to select a **type graph** $H_{\text{typ}} = (U, B, F)$, a **known** distribution $\mathcal{D}$ supported on B, and the number of arrivals $n \geq 1$.
- Online vertices $v_1, \ldots, v_n$ are drawn independently from $\mathcal{D}$, and presented to the algorithm one by one.

Online Stochastic Matching

- Since **1/e** is best possible, various works have focused on understanding even more optimistic online models than ROM.
- A popular setting is the **prophet matching problem with known i.i.d. arrivals** which considers when $G = (U, V, E)$ is drawn from a distribution.
- The adversary now only gets to select a **type graph** $H_{\text{typ}} = (U, B, F)$, a **known** distribution $\mathcal{D}$ supported on B, and the number of arrivals $n \geq 1$.
- Online vertices $v_1, \ldots, v_n$ are drawn independently from $\mathcal{D}$, and presented to the algorithm one by one.
- Both the algorithm and the benchmark average their performance over G drawn from $\mathcal{D}$.

Online Stochastic Matching

- Since **1/e** is best possible, various works have focused on understanding even more optimistic online models than ROM.
- A popular setting is the **prophet matching problem with known i.i.d. arrivals** which considers when $G = (U, V, E)$ is drawn from a distribution.
- The adversary now only gets to select a **type graph** $H_{\text{typ}} = (U, B, F)$, a **known** distribution $\mathcal{D}$ supported on B, and the number of arrivals $n \geq 1$.
- Online vertices $v_1, \ldots, v_n$ are drawn independently from $\mathcal{D}$, and presented to the algorithm one by one.
- Both the algorithm and the benchmark average their performance over G drawn from $\mathcal{D}$.
- A competitive ratio of **1-1/e** is attainable due to Manshadi et al. (2012), and this is the best known result for edge weights.

Open Problems

- For a single offline item (i.e., $|U| = 1$), $\approx$ **0.745** is attainable, and this known to be tight.

Open Problems

- For a single offline item (i.e., $|U| = 1$), $\approx$ **0.745** is attainable, and this known to be tight. This special case is the famous **prophet inequality problem** for i.i.d. random variables.

Open Problems

- For a single offline item (i.e., $|U| = 1$), $\approx$ **0.745** is attainable, and this known to be tight. This special case is the famous **prophet inequality problem** for i.i.d. random variables.
- What is the best possible competitive ratio for the prophet matching problem with known i.i.d. arrivals?

Open Problems

- For a single offline item (i.e., $|U| = 1$), $\approx$ **0.745** is attainable, and this known to be tight. This special case is the famous **prophet inequality problem** for i.i.d. random variables.
- What is the best possible competitive ratio for the prophet matching problem with known i.i.d. arrivals? Can **1-1/e** be beaten as in the single item setting?

Open Problems

- For a single offline item (i.e., $|U| = 1$), $\approx$ **0.745** is attainable, and this known to be tight. This special case is the famous **prophet inequality problem** for i.i.d. random variables.
- What is the best possible competitive ratio for the prophet matching problem with known i.i.d. arrivals? Can **1-1/e** be beaten as in the single item setting?
- There are numerous works answering this in the affirmative for special distributions, and/or simpler type graphs.