Formal Verification of the RANKING algorithm for Online Bipartite Matching

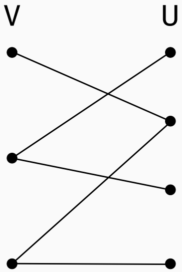
Christoph Madlener

22.06.2022

Online Bipartite Matching (OBM)

Input

- *bipartite* graph $G = (U, V, E)$



Online Bipartite Matching (OBM)

Input

- *bipartite* graph $G = (U, V, E)$
- *offline* vertices V are known

Online Bipartite Matching (OBM)

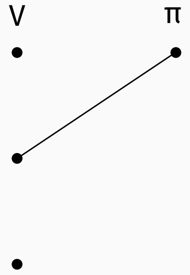
Input

- *bipartite* graph $G = (U, V, E)$
- *offline* vertices V are known
- *online* vertices U reveal edges on arrival

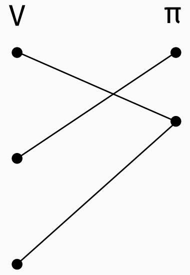
Online Bipartite Matching (OBM)

Input

- *bipartite* graph $G = (U, V, E)$
- *offline* vertices V are known
- *online* vertices U reveal edges on arrival



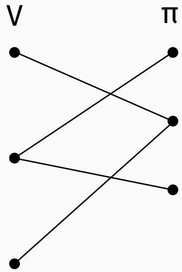
Online Bipartite Matching (OBM)



Input

- *bipartite* graph $G = (U, V, E)$
- *offline* vertices V are known
- *online* vertices U reveal edges on arrival

Online Bipartite Matching (OBM)



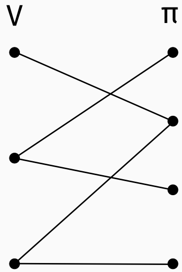
Input

- *bipartite* graph $G = (U, V, E)$
- *offline* vertices V are known
- *online* vertices U reveal edges on arrival

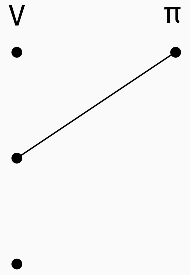
Online Bipartite Matching (OBM)

Input

- *bipartite* graph $G = (U, V, E)$
- *offline* vertices V are known
- *online* vertices U reveal edges on arrival



Online Bipartite Matching (OBM)



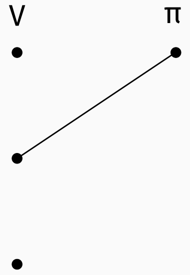
Input

- *bipartite* graph $G = (U, V, E)$
- *offline* vertices V are known
- *online* vertices U reveal edges on arrival

Task

- on arrival of $u \in U$, match to *unmatched* neighbor $v \in V$ (or not)

Online Bipartite Matching (OBM)



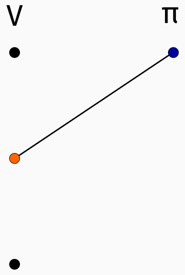
Input

- *bipartite* graph $G = (U, V, E)$
- *offline* vertices V are known
- *online* vertices U reveal edges on arrival

Task

- on arrival of $u \in U$, match to *unmatched* neighbor $v \in V$ (or not)
- maximize size of resulting matching

Online Bipartite Matching (OBM)



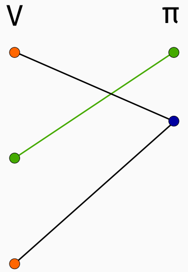
Input

- *bipartite* graph $G = (U, V, E)$
- *offline* vertices V are known
- *online* vertices U reveal edges on arrival

Task

- on arrival of $u \in U$, match to *unmatched* neighbor $v \in V$ (or not)
- maximize size of resulting matching

Online Bipartite Matching (OBM)



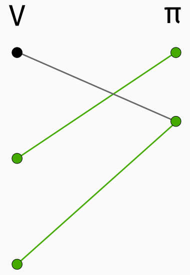
Input

- *bipartite* graph $G = (U, V, E)$
- *offline* vertices V are known
- *online* vertices U reveal edges on arrival

Task

- on arrival of $u \in U$, match to *unmatched* neighbor $v \in V$ (or not)
- maximize size of resulting matching

Online Bipartite Matching (OBM)



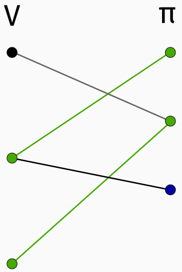
Input

- *bipartite* graph $G = (U, V, E)$
- *offline* vertices V are known
- *online* vertices U reveal edges on arrival

Task

- on arrival of $u \in U$, match to *unmatched* neighbor $v \in V$ (or not)
- maximize size of resulting matching

Online Bipartite Matching (OBM)



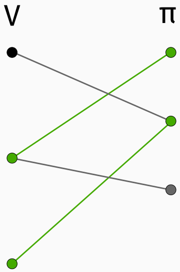
Input

- *bipartite* graph $G = (U, V, E)$
- *offline* vertices V are known
- *online* vertices U reveal edges on arrival

Task

- on arrival of $u \in U$, match to *unmatched* neighbor $v \in V$ (or not)
- maximize size of resulting matching

Online Bipartite Matching (OBM)



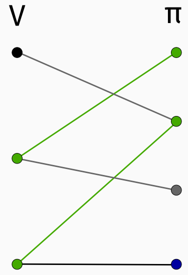
Input

- *bipartite* graph $G = (U, V, E)$
- *offline* vertices V are known
- *online* vertices U reveal edges on arrival

Task

- on arrival of $u \in U$, match to *unmatched* neighbor $v \in V$ (or not)
- maximize size of resulting matching

Online Bipartite Matching (OBM)



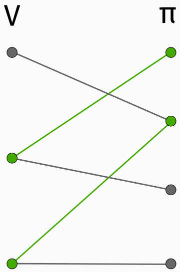
Input

- *bipartite* graph $G = (U, V, E)$
- *offline* vertices V are known
- *online* vertices U reveal edges on arrival

Task

- on arrival of $u \in U$, match to *unmatched* neighbor $v \in V$ (or not)
- maximize size of resulting matching

Online Bipartite Matching (OBM)



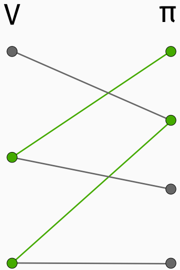
Input

- *bipartite* graph $G = (U, V, E)$
- *offline* vertices V are known
- *online* vertices U reveal edges on arrival

Task

- on arrival of $u \in U$, match to *unmatched* neighbor $v \in V$ (or not)
- maximize size of resulting matching

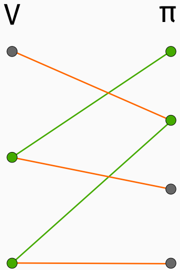
Competitive Ratio



Performance of online algorithm $\mathcal{A}$

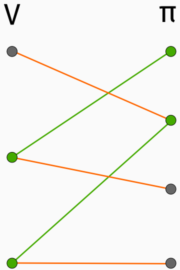
- Compare $\mathcal{A}$ to best offline algorithm

Competitive Ratio



Performance of online algorithm $\mathcal{A}$

- Compare $\mathcal{A}$ to best offline algorithm



Performance of online algorithm $\mathcal{A}$

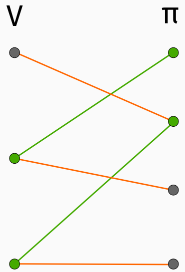
- Compare $\mathcal{A}$ to best offline algorithm

Competitive ratio for OBM

$$\min_G \min_\pi \frac{|\mathcal{A}(G,\pi)|}{|M|}$$

where M is a maximum cardinality matching in G.

Competitive Ratio



Performance of online algorithm $\mathcal{A}$

- Compare $\mathcal{A}$ to best offline algorithm

Competitive ratio for OBM

$$\min_{G} \min_{\pi} \frac{\mathbb{E}\big[|\mathcal{A}(G,\pi)|\big]}{|M|}$$

where M is a maximum cardinality matching in G.

RANKING

- simple randomized algorithm due to Karp, Vazirani, and Vazirani is optimal [KVV90]

- simple randomized algorithm due to Karp, Vazirani, and Vazirani is optimal [KVV90]

Algorithm 1: RANKING

Initialization: Choose a random permutation (ranking) σ of V
Online Matching:
On arrival of $u \in U$
    $N(u) \leftarrow$ set of unmatched neighbors of u
    if $N(u) \neq \emptyset$
        match u to the vertex $v \in N(u)$ that minimizes $\sigma(v)$

RANKING

- simple randomized algorithm due to Karp, Vazirani, and Vazirani is optimal [KVV90]

Algorithm 1: RANKING

Initialization: Choose a random permutation (ranking) σ of V
Online Matching:
On arrival of $u \in U$
  $N(u) \leftarrow$ set of unmatched neighbors of u
  if $N(u) \neq \emptyset$
    match u to the vertex $v \in N(u)$ that minimizes $\sigma(v)$

- competitive ratio of $1 - \frac{1}{e}$ (best possible)

Formalization Outline

- formalization follows proof due to Birnbaum & Mathieu [BM08]

Formalization Outline

- formalization follows proof due to Birnbaum & Mathieu [BM08]
- three parts

Formalization Outline

- formalization follows proof due to Birnbaum & Mathieu [BM08]
- three parts
    1. Combinatorics

Formalization Outline

- formalization follows proof due to Birnbaum & Mathieu [BM08]
- three parts
  1. Combinatorics
  2. Probability theory

Formalization Outline

- formalization follows proof due to Birnbaum & Mathieu [BM08]
- three parts
    1. Combinatorics
    2. Probability theory
    3. Competitive ratio in the limit

Formalization Outline

- formalization follows proof due to Birnbaum & Mathieu [BM08]
- three parts
  1. **Combinatorics**
  2. Probability theory
  3. Competitive ratio in the limit

Reducing Analysis to Graphs with Perfect Matching

- original paper (and earlier simplifications) assume G has a perfect matching

Reducing Analysis to Graphs with Perfect Matching

- original paper (and earlier simplifications) assume G has a perfect matching
- Birnbaum & Mathieu state a *simple structural observation* which allows to generalize to arbitrary graphs:

Lemma 2. *Let x be a vertex, $H = G \setminus \{x\}$, and π_H and σ_H be the orderings of U_H and V_H induced by π and σ respectively. If the matchings Ranking(H, π_H, σ_H) and Ranking(G, π, σ) are not identical, then they differ by a single alternating path starting at vertex x.*

Reducing Analysis to Graphs with Perfect Matching

- original paper (and earlier simplifications) assume G has a perfect matching
- Birnbaum & Mathieu state a *simple structural observation* which allows to generalize to arbitrary graphs:

Lemma 2. *Let x be a vertex, $H = G \setminus \{x\}$, and π_H and σ_H be the orderings of U_H and V_H induced by π and σ respectively. If the matchings $Ranking(H, \pi_H, \sigma_H)$ and $Ranking(G, \pi, \sigma)$ are not identical, then they differ by a single alternating path starting at vertex x.*

Let G_i be the graph resulting from removing i vertices from G, which are not in a maximum cardinality matching M, and $R_i := Ranking(H_i, \pi_{H_i}, \sigma_{H_i})$.

Reducing Analysis to Graphs with Perfect Matching

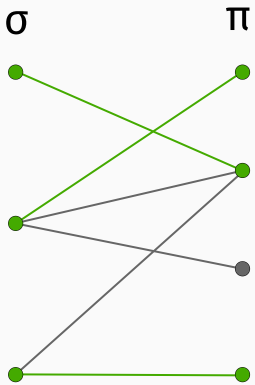
- original paper (and earlier simplifications) assume G has a perfect matching
- Birnbaum & Mathieu state a *simple structural observation* which allows to generalize to arbitrary graphs:

Lemma 2. *Let x be a vertex, $H = G \setminus \{x\}$, and π_H and σ_H be the orderings of U_H and V_H induced by π and σ respectively. If the matchings Ranking(H, π_H, σ_H) and Ranking(G, π, σ) are not identical, then they differ by a single alternating path starting at vertex x.*

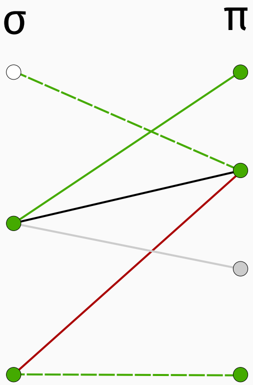
Let G_i be the graph resulting from removing i vertices from G, which are not in a maximum cardinality matching M, and $R_i := Ranking(H_i, \pi_{H_i}, \sigma_{H_i})$.

$$\frac{|Ranking(G, \pi, \sigma)|}{|M|} \geq \frac{|R_1|}{|M|} \geq \dots \geq \frac{|Ranking(G^*, \pi_{G^*}, \sigma_{G^*})|}{|M|}$$

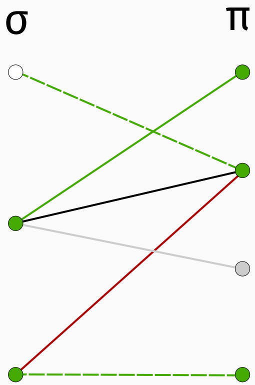
First proof of a *simple structural observation*



First proof of a *simple structural observation*



First proof of a *simple structural observation*



Let $R := Ranking(G, \pi, \sigma)$ for a fixed graph G, arrival order π, and ranking σ.

Specification of alternating path

$$zig(x) = \begin{cases} x \# zag(y) & \{x, y\} \in R \\ [x] & x \text{ unmatched} \end{cases}$$

$$zag(y) = \begin{cases} y \# zag(x') & x' \text{ matched instead} \\ [y] & \text{no other match} \end{cases}$$

Randomization

- rephrase everything as _ *pmf* (probability mass function)

Randomization

- rephrase everything as _ *pmf* (probability mass function)
- finite probability spaces over permutations → lots of sums

Randomization

- rephrase everything as _ *pmf* (probability mass function)
- finite probability spaces over permutations → lots of sums

Switching probability spaces

Randomization

- rephrase everything as _ *pmf* (probability mass function)
- finite probability spaces over permutations → lots of sums

Switching probability spaces

1. choosing a random permutation **vs.**

Randomization

- rephrase everything as _ *pmf* (probability mass function)
- finite probability spaces over permutations → lots of sums

Switching probability spaces

1. choosing a random permutation **vs.**
2. choosing a random permutation, a random vertex, and putting that vertex at index t

Randomization

- rephrase everything as _ *pmf* (probability mass function)
- finite probability spaces over permutations → lots of sums

Switching probability spaces

1. choosing a random permutation **vs.**
2. choosing a random permutation, a random vertex, and putting that vertex at index t

For $t = 1$ and $V = \{1, 2, 3\}$:

$$\mathbb{P}_1\Big(\big\{[3, 2, 1]\big\}\Big) = \frac{1}{3!}$$

$$\begin{aligned}\mathbb{P}_2\Big(\big\{[3, 2, 1]\big\}\Big) &= \mathbb{P}_1\Big(\big\{[2, 3, 1], [3, 1, 2], [3, 2, 1]\big\}\Big) \cdot \mathbb{P}_V(\{2\}) \\ &= \frac{3}{3!} \cdot \frac{1}{3} = \frac{1}{3!}\end{aligned}$$

References

Benjamin Birnbaum and Claire Mathieu.
On-line bipartite matching made simple.
Acm Sigact News, 39(1):80–87, 2008.

Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani.
An optimal algorithm for on-line bipartite matching.
In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 352–358, 1990.

Christoph Madlener.
Formal Verification of the RANKING Algorithm for Online Bipartite Matching.
`https://github.com/cmadlener/isabelle-ranking`, 2022.