Formal Verification of the RANKING algorithm for Online Bipartite Matching

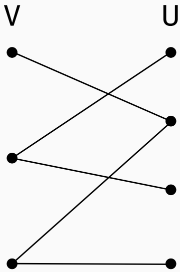
Christoph Madlener

22.06.2022

Online Bipartite Matching (OBM)

Input

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



Online Bipartite Matching (OBM)

Input

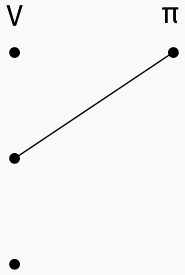
- *bipartite* graph $G = (U, V, E)$
- *offline* vertices V are known

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)

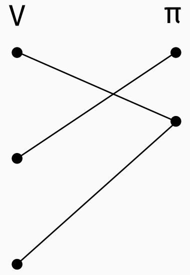


Input

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

Input

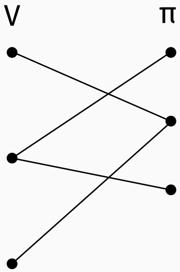
- *bipartite* graph $G = (U, V, E)$
- *offline* vertices V are known
- *online* vertices U reveal edges on arrival



Online Bipartite Matching (OBM)

Input

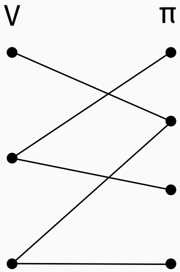
- *bipartite* graph $G = (U, V, E)$
- *offline* vertices V are known
- *online* vertices U reveal edges on arrival



Online Bipartite Matching (OBM)

Input

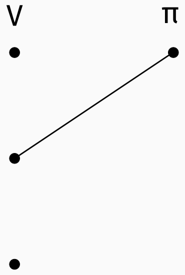
- *bipartite* graph $G = (U, V, E)$
- *offline* vertices V are known
- *online* vertices U reveal edges on arrival



Online Bipartite Matching (OBM)

Input

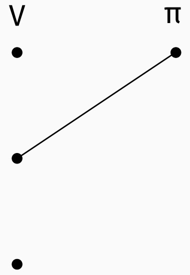
- *bipartite* graph $G = (U, V, E)$
- *offline* vertices V are known
- *online* vertices U reveal edges on arrival



Task

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

Online Bipartite Matching (OBM)



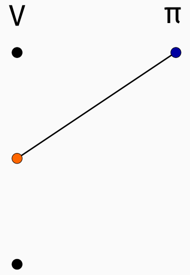
Input

- *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)



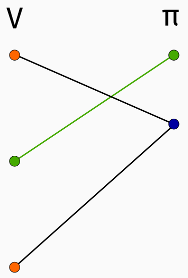
Input

- *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)



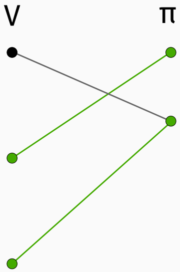
Input

- *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)



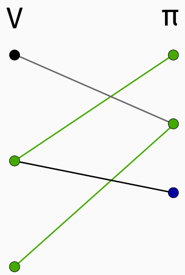
Input

- *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)



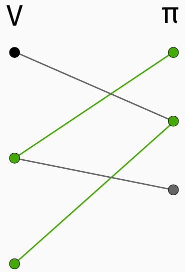
Input

- *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)



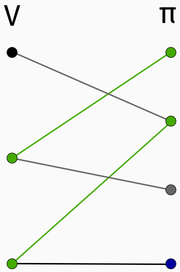
Input

- *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)



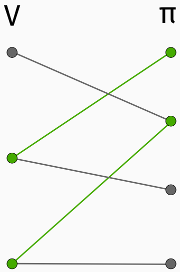
Input

- *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)



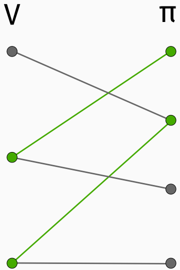
Input

- *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

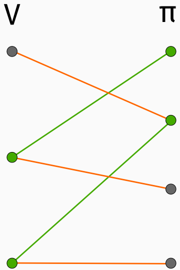
Competitive Ratio



Performance of online algorithm $\mathcal{A}$

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

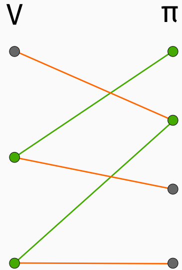
Competitive Ratio



Performance of online algorithm $\mathcal{A}$

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

Competitive Ratio



Performance of online algorithm $\mathcal{A}$

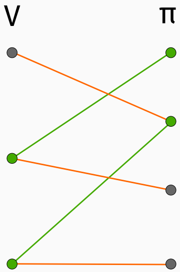
- Compare $\mathcal{A}$ to best offline algorithm

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

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

where M is a maximum cardinality matching in G.

RANKING

- simple randomized algorithm due to Karp, Vazirani, and Vazirani is optimal [3]

RANKING

- simple randomized algorithm due to Karp, Vazirani, and Vazirani is optimal [3]

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 $\sigma(v)$

- simple randomized algorithm due to Karp, Vazirani, and Vazirani is optimal [3]

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, and Mathieu [2]

Formalization Outline

- formalization follows proof due to Birnbaum, and Mathieu [2]
- two *sections*

Formalization Outline

- formalization follows proof due to Birnbaum, and Mathieu [2]
- two *sections*
    1. Combinatorics

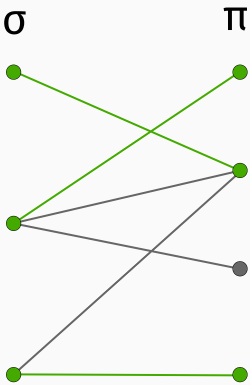
Formalization Outline

- formalization follows proof due to Birnbaum, and Mathieu [2]
- two *sections*
    1. Combinatorics
    2. Probability theory

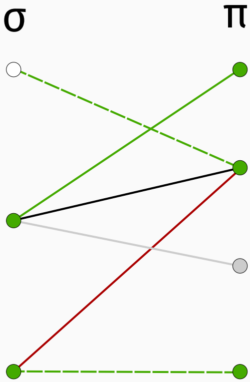
Formalization Outline

- formalization follows proof due to Birnbaum, and Mathieu [2]
- two *sections*
  1. **Combinatorics**
  2. Probability theory
- Most involved part: *Lemma 2*
  "when removing a vertex x from the graph, then the runs on the original graph, and the one without x, differ by at most one alternating path, starting at x"

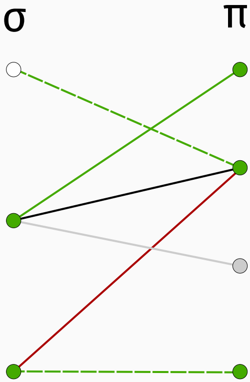
A ”simple structural observation”



A ”simple structural observation”



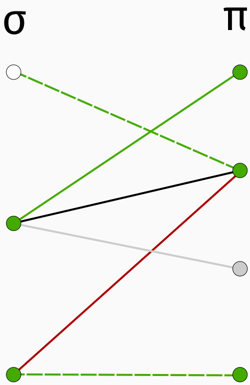
A ”simple structural observation”



Keys to formal proof

- *non-recursive* specification of matching on G with arrival order π, and ranking σ

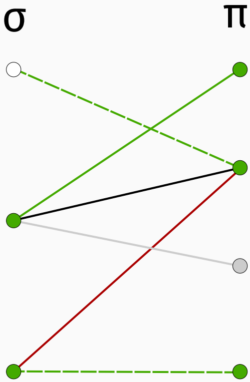
A ”simple structural observation”



Keys to formal proof

- *non-recursive* specification of matching on G with arrival order π, and ranking σ
- gives interchangebility of offline and online vertices

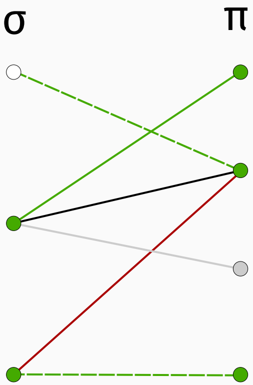
A ”simple structural observation”



Keys to formal proof

- *non-recursive* specification of matching on G with arrival order π, and ranking σ
- gives interchangebility of offline and online vertices
- full specification of path with mutually recursive functions *zig* and *zag*

A ”simple structural observation”



Keys to formal proof

- *non-recursive* specification of matching on G with arrival order π, and ranking σ
- gives interchangebility of offline and online vertices
- full specification of path with mutually recursive functions *zig* and *zag*
- Berge's Lemma formalized by Abdulaziz [1]

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

- choosing a random permutation **vs.**
  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

- choosing a random permutation **vs.**
  choosing a random permutation, a random vertex, and putting that vertex at index t
- choosing a random permutation of the original offline vertices **vs.**
  choosing a random permutation of the *reduced* offline vertices

References

M. Abdulaziz, K. Mehlhorn, and T. Nipkow.
Trustworthy graph algorithms.
arXiv preprint arXiv:1907.04065, 2019.

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

R. M. Karp, U. V. Vazirani, and V. 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.