Formal Verification of the RANKING algorithm for Online Bipartite Matching

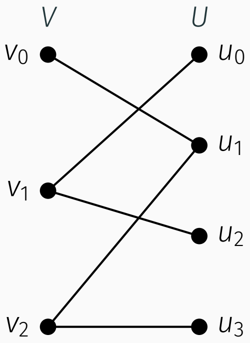
Christoph Madlener

22.06.2022

Online Bipartite Matching (OBM)

Input

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

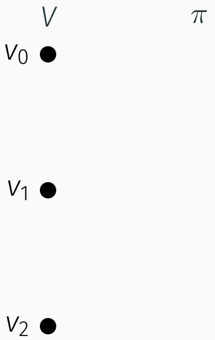


Online Bipartite Matching (OBM)

Input

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

Online Bipartite Matching (OBM)



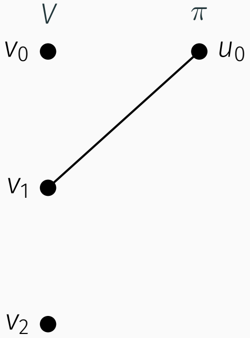
Input

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

Online Bipartite Matching (OBM)

Input

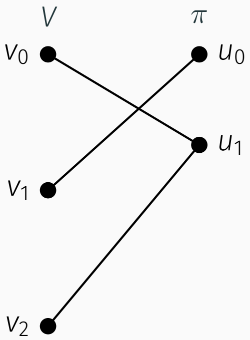
- *bipartite* graph $G = (U, V, E)$
- *offline* vertices V are known
- *online* vertices U reveal edges on arrival



Online Bipartite Matching (OBM)

Input

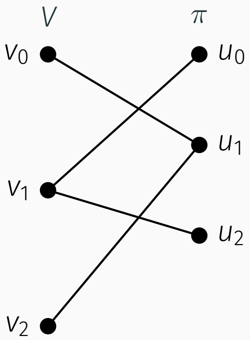
- *bipartite* graph $G = (U, V, E)$
- *offline* vertices V are known
- *online* vertices U reveal edges on arrival



Online Bipartite Matching (OBM)

Input

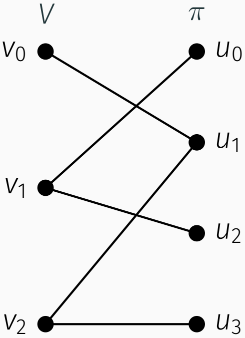
- *bipartite* graph $G = (U, V, E)$
- *offline* vertices V are known
- *online* vertices U reveal edges on arrival



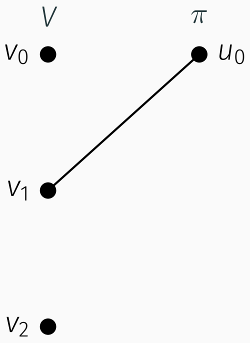
Online Bipartite Matching (OBM)

Input

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



Online Bipartite Matching (OBM)



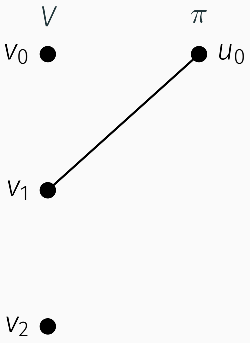
Input

- *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)



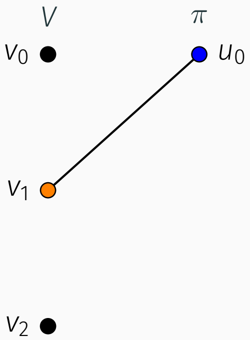
Input

- *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)



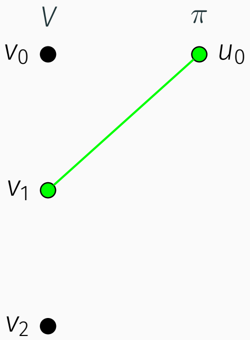
Input

- *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)



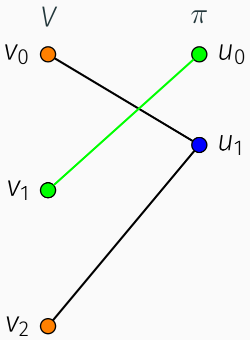
Input

- *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)



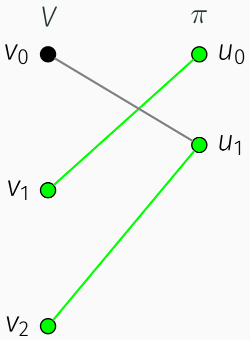
Input

- *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)



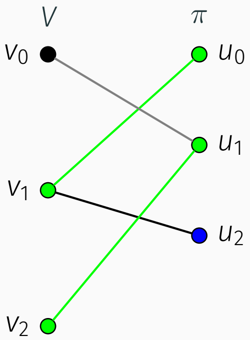
Input

- *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)



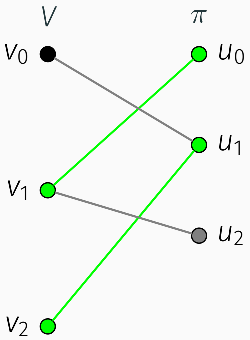
Input

- *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)



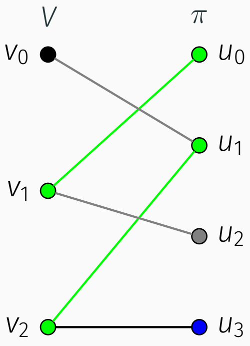
Input

- *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)



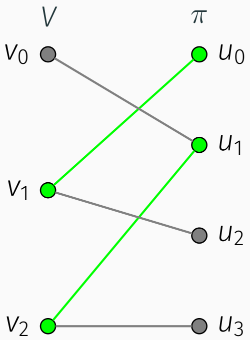
Input

- *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

Online Bipartite Matching (OBM)



Input

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

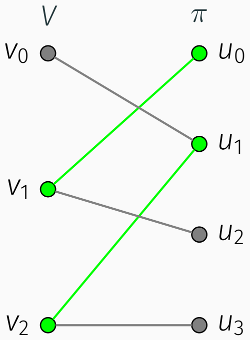
Task

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

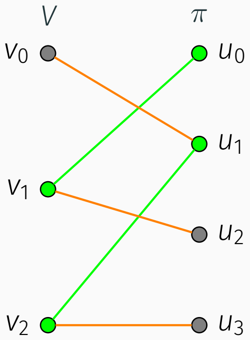
Competitive Ratio

Performance of online algorithm $\mathcal{A}$

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



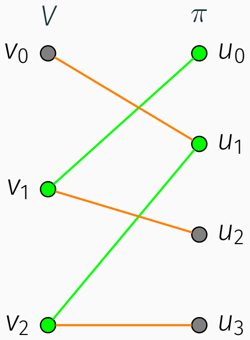
Competitive Ratio



Performance of online algorithm $\mathcal{A}$

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

Competitive Ratio



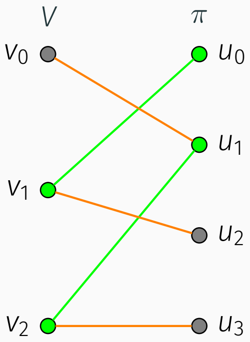
Performance of online algorithm $\mathcal{A}$

- 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.



Performance of online algorithm $\mathcal{A}$

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

Competitive ratio for OBM

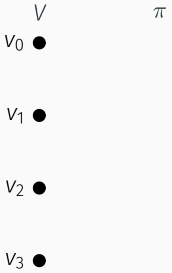
$$\min_G \min_\pi \frac{\mathbb{E}\left[|\mathcal{A}(G,\pi)|\right]}{|M|}$$

where M is a maximum cardinality matching in G.

RANKING

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

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



Initialization: Choose a random permutation (ranking) σ of V

Online Matching:

On arrival of $u \in U$

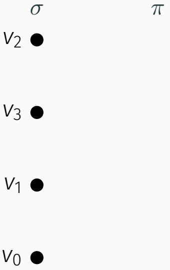
  $N(u) \leftarrow$ set of unmatched neighbors of u

  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]



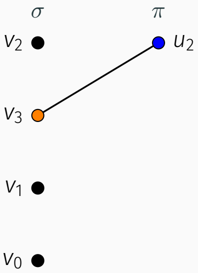
Initialization: Choose a random permutation (ranking) σ of V

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]



Initialization: Choose a random permutation (ranking) σ of V

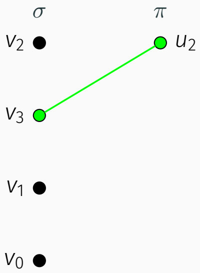
Online Matching:

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]



Initialization: Choose a random permutation (ranking) σ of V

Online Matching:

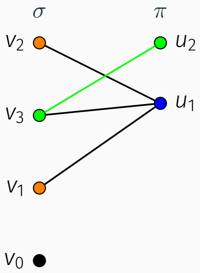
On arrival of $u \in U$

  $N(u) \leftarrow$ 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 [KVV90]



Initialization: Choose a random permutation (ranking) σ of V

Online Matching:

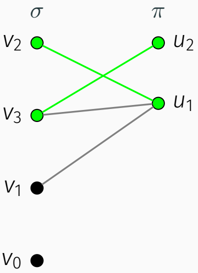
On arrival of $u \in U$

$N(u) \leftarrow$ 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 [KVV90]



Initialization: Choose a random permutation (ranking) σ of V

Online Matching:

On arrival of $u \in U$

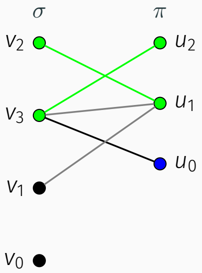
  $N(u) \leftarrow$ set of unmatched neighbors of u

  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]



Initialization: Choose a random permutation (ranking) σ of V

Online Matching:

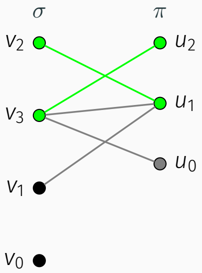
On arrival of $u \in U$

  $N(u) \leftarrow$ 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 [KVV90]



Initialization: Choose a random permutation (ranking) σ of V

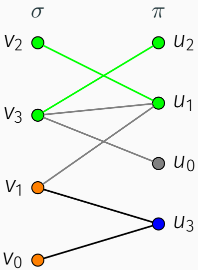
Online Matching:

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]



Initialization: Choose a random permutation (ranking) σ of V

Online Matching:

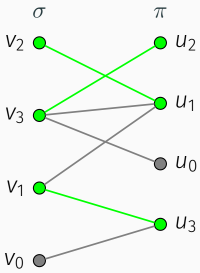
On arrival of $u \in U$

  $N(u) \leftarrow$ 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 [KVV90]



Initialization: Choose a random permutation (ranking) σ of V

Online Matching:

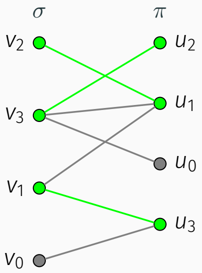
On arrival of $u \in U$

  $N(u) \leftarrow$ 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 [KVV90]



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

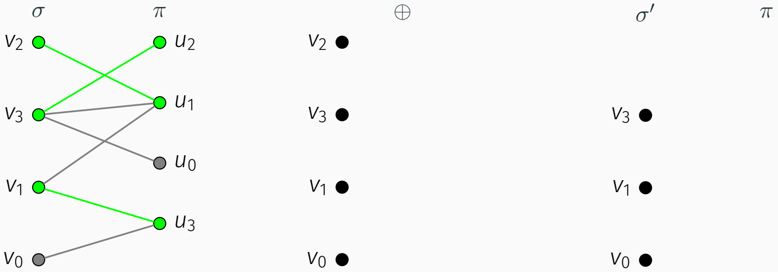
- original paper (and earlier simplifications) assume G has a perfect matching

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:

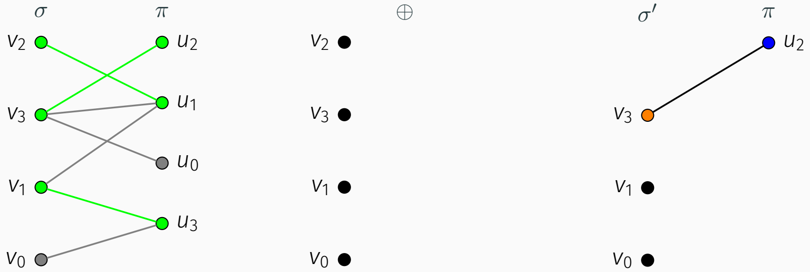
Reducing Analysis to Graphs with Perfect Matching

- 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:



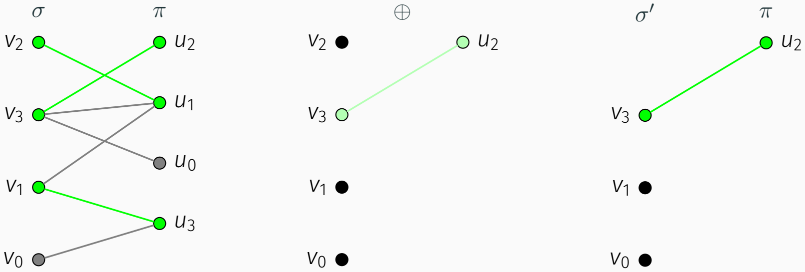
Reducing Analysis to Graphs with Perfect Matching

- 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:



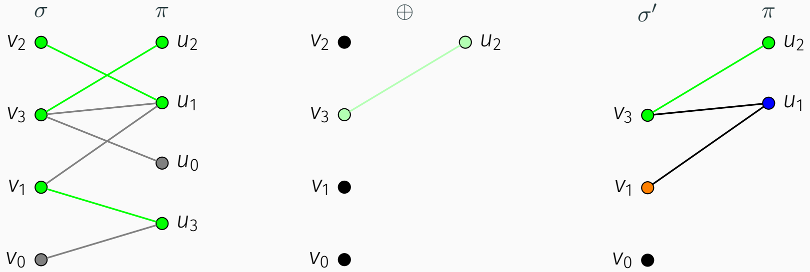
Reducing Analysis to Graphs with Perfect Matching

- 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:



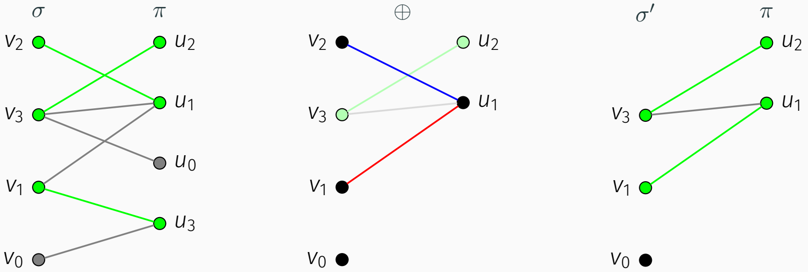
Reducing Analysis to Graphs with Perfect Matching

- 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:



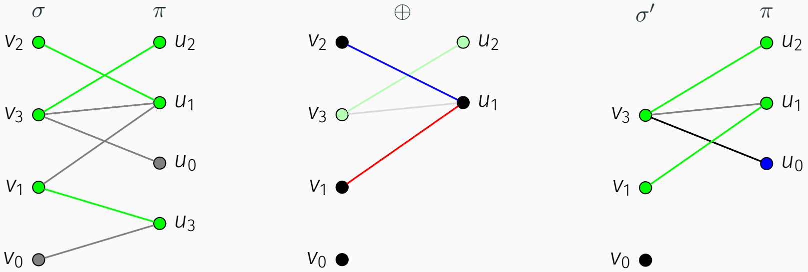
Reducing Analysis to Graphs with Perfect Matching

- 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:



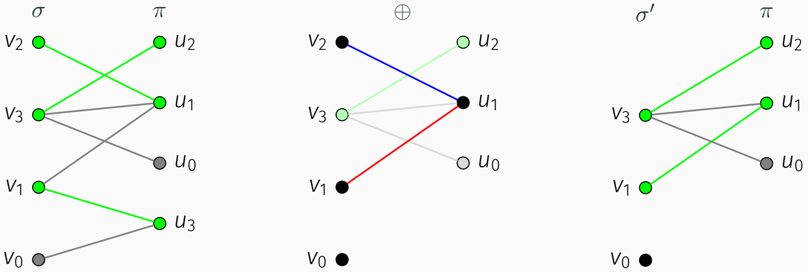
Reducing Analysis to Graphs with Perfect Matching

- 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:



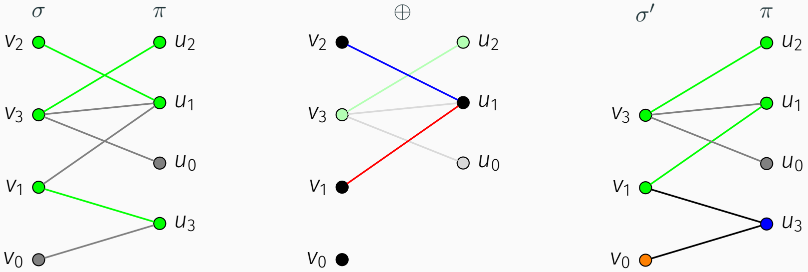
Reducing Analysis to Graphs with Perfect Matching

- 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:



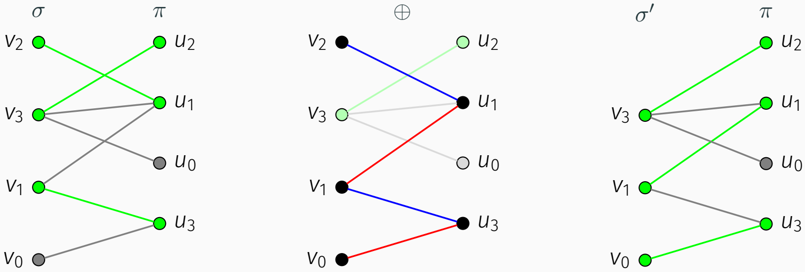
Reducing Analysis to Graphs with Perfect Matching

- 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:



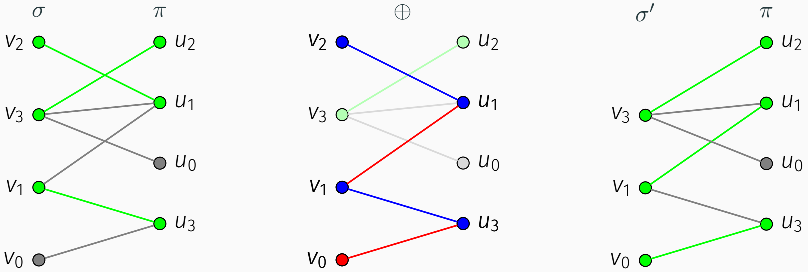
Reducing Analysis to Graphs with Perfect Matching

- 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:

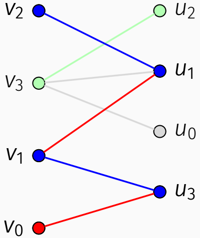


Reducing Analysis to Graphs with Perfect Matching

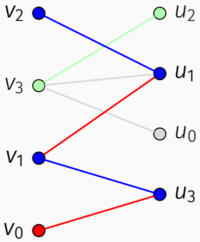
- original paper (and earlier simplifications) assume G has a perfect matching
- Birnbaum & Mathieu state a *simple structural observation* which allows to generalize to arbitrary graphs:



First proof of a *simple structural observation*



First proof of a *simple structural observation*



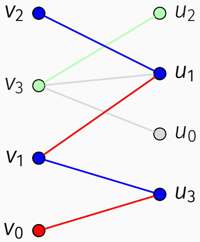
Let $R := \mathit{Ranking}(G, \pi, \sigma)$ for a fixed graph G, arrival order π, and ranking σ.

Specification of alternating path

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

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

First proof of a *simple structural observation*



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

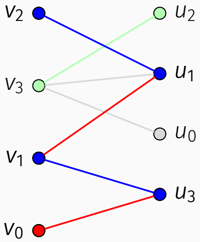
Specification of alternating path

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

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

- non-recursive specification of $Ranking(G, \pi, \sigma)$
  $\rightarrow$ interchangeability of U, V

First proof of a *simple structural observation*



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

Specification of alternating path

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

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

- non-recursive specification of $\mathit{Ranking}(G, \pi, \sigma)$
  $\rightarrow$interchangeability of U, V
- Berge's Lemma [AMN19] for repeated application

Randomization

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

Randomization

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

- 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 $V = \{1, 2, 3\}$:

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

$$\mathbb{P}_2\Big(\{[3,2,1]\}\Big) = \mathbb{P}_1\Big(\{[2,3,1],[3,1,2],[3,2,1]\}\Big) \cdot \mathbb{P}_V(\{2\})$$

$$= \frac{3}{3!} \cdot \frac{1}{3} = \frac{1}{3!}$$

References

Mohammad Abdulaziz, Kurt Mehlhorn, and Tobias Nipkow.
Trustworthy graph algorithms.
arXiv preprint arXiv:1907.04065, 2019.

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`,