

Stable Matchings

Meghana Nasre

IIT Madras

3rd Summer School on Theoretical Foundations of Computer Science 2021

IIIT Bangalore

July 15, 2021

Problem Setup

- A set of men \mathcal{A} (applicants / students / medical interns)
- A set of women \mathcal{B} (jobs / courses / hospitals)
- Each participant has a preference ordering.

a_1 : b_1 b_2
 a_2 : b_1 b_2

b_1 : a_1 a_2
 b_2 : a_1 a_2

- Here preferences are **strict** and **complete**.

Goal: Assign men to women **optimally**.

Problem Setup

- A set of men \mathcal{A} (applicants / students / medical interns)
- A set of women \mathcal{B} (jobs / courses / hospitals)
- Each participant has a preference ordering.

a_1 : b_1 b_2
 a_2 : b_1 b_2

b_1 : a_1 a_2
 b_2 : a_1 a_2

- Here preferences are **strict** and **complete**.

A possible assignment $M = \{(a_1, b_2), (a_2, b_1)\}$.

Stability – a notion of optimality

a_1 :	b_1	b_2
a_2 :	b_1	b_2

b_1 :	a_1	a_2
b_2 :	a_1	a_2

A pair $(a, b) \in E \setminus M$ **blocks** M if

- Both a and b prefer each other to their current partner in M .
- (a_1, b_1) : both a_1 and b_1 wish to deviate – **blocking pair**.

A matching is **stable** if no pair wishes to deviate.

$M' = \{(a_1, b_1), (a_2, b_2)\}$ is a stable.

Known Facts:

- Every instance admits a stable matching.
- Stable matching can be computed in linear time.
- All stable matchings are **perfect**.

Today and tomorrow: Two Variants of the SM problem

Variation #1: Incomplete Lists

- Preferences are **strict** and can be **incomplete**.

a_1 : b_1 b_2
 a_2 : b_1

b_1 : a_1 a_2
 b_2 : a_1

- Does a stable matching exist? **Yes!** $M = \{(a_1, b_1)\}$.

Known Facts:

- Every instance admits a stable matching; can be computed in linear time.
- All stable matchings are **perfect** of **the same size**.
- Stable matching can be **half** the size of max. matching.

Question: Are there larger optimal matchings?

Variation #2: Incomplete Lists and Ties

- Preferences can contain **ties** and can be **incomplete**.

$$\begin{array}{lll}
 a_1: & b_1 & b_2 \\
 a_2: & b_1 & \\
 \end{array}
 \qquad
 \begin{array}{ll}
 b_1: & (a_1 \ a_2) \\
 b_2: & a_1
 \end{array}$$

- Redefine blocking pair.
 - A pair $(a, b) \in E \setminus M$ **blocks** M if Both a and b **strictly** prefer each other to their current partner in M .

- Does a stable matching exist? **Yes!**

$$M_1 = \{(a_1, b_1)\} \qquad M_2 = \{(a_1, b_2), (a_2, b_1)\}$$

Known Facts:

- Every instance admits a stable matching; can be computed in linear time.
- All stable matchings ~~are of the same size~~ **need not be of same size**.
- A stable matching can be **half** the size of another stable matching.

Question: How to compute largest size stable matching?

Classical Model: Strict and Complete lists

Computing a stable matching

Gale and Shapley 1962

a_1 : b_1 b_2
 a_2 : b_1 b_2

b_1 : a_1 a_2
 b_2 : a_1 a_2

Computing a stable matching

Gale and Shapley 1962

a_1 : b_1 b_2
 a_2 : b_1 b_2

b_1 : a_1 a_2
 b_2 : a_1 a_2

Gale and Shapley Algo.

- Men propose.
- Women accept / reject.

Computing a stable matching

Gale and Shapley 1962

a_1 : b_1 b_2
 a_2 : b_1 b_2

b_1 : a_1 a_2
 b_2 : a_1 a_2

Gale and Shapley Algo.

- Men propose.
- Women accept / reject.

■ $a_1 \rightarrow b_1$

Computing a stable matching

Gale and Shapley 1962

a_1 : b_1 b_2
 a_2 : b_1 b_2

b_1 : a_1 a_2
 b_2 : a_1 a_2

Gale and Shapley Algo.

- Men propose.
- Women accept / reject.

- $a_1 \rightarrow b_1$ accept.

Computing a stable matching

Gale and Shapley 1962

a_1 : b_1 b_2
 a_2 : b_1 b_2

b_1 : a_1 a_2
 b_2 : a_1 a_2

Gale and Shapley Algo.

- Men propose.
 - Women accept / reject.
- $a_1 \rightarrow b_1$ accept.
 - $a_2 \rightarrow b_1$

Computing a stable matching

Gale and Shapley 1962

a_1 : $\boxed{b_1}$ b_2
 a_2 : b_1 b_2

b_1 : $\boxed{a_1}$ a_2
 b_2 : a_1 a_2

Gale and Shapley Algo.

- Men propose.
 - Women accept / reject.
- $a_1 \rightarrow b_1$ accept.
 - $a_2 \rightarrow b_1$ reject.

Computing a stable matching

Gale and Shapley 1962

a_1 : b_1 b_2
 a_2 : b_1 b_2

b_1 : a_1 a_2
 b_2 : a_1 a_2

Gale and Shapley Algo.

- Men propose.
- Women accept / reject.
- $a_1 \rightarrow b_1$ accept.
- $a_2 \rightarrow b_1$ reject.
- $a_2 \rightarrow b_2$ accept.

Computing a stable matching

Gale and Shapley 1962

a_1 : $\boxed{b_1}$ b_2
 a_2 : b_1 $\boxed{b_2}$

b_1 : $\boxed{a_1}$ a_2
 b_2 : a_1 $\boxed{a_2}$

Gale and Shapley Algo.

- Men propose.
 - Women accept / reject.
- $a_1 \rightarrow b_1$ accept.
 - $a_2 \rightarrow b_1$ reject.
 - $a_2 \rightarrow b_2$ accept.
-

Computing a stable matching

Gale and Shapley 1962

a_1 : $\boxed{b_1}$ b_2
 a_2 : b_1 $\boxed{b_2}$

b_1 : $\boxed{a_1}$ a_2
 b_2 : a_1 $\boxed{a_2}$

Gale and Shapley Algo.

- Men propose.
- Women accept / reject.

- $a_1 \rightarrow b_1$ accept.
- $a_2 \rightarrow b_1$ reject.
- $a_2 \rightarrow b_2$ accept.

$$M_s = \{(a_1, b_1), (a_2, b_2)\}.$$

- Order of proposals does not matter.
- The side which proposes does matter.

Models : Recap

Model	Details	Goal	
Classical setting	strict and complete list	Compute a stable matching	✓
Variation #1	strict and incomplete list	Compute a larger optimal matching	
Variation #2	ties and incomplete list	Compute a largest stable matching	

Variation #2: Incomplete Lists and Ties

Variation #2: Incomplete Lists and Ties

Assume ties only on B side.

a_1 : b_1 b_2

a_2 : b_1

b_1 : (a_1 a_2)

b_2 : a_1

Recall:

- Multiple stable matchings of different sizes.

$$M_1 = \{(a_1, b_1)\} \quad M_2 = \{(a_1, b_2), (a_2, b_1)\}$$

- Compute **largest** size stable matching

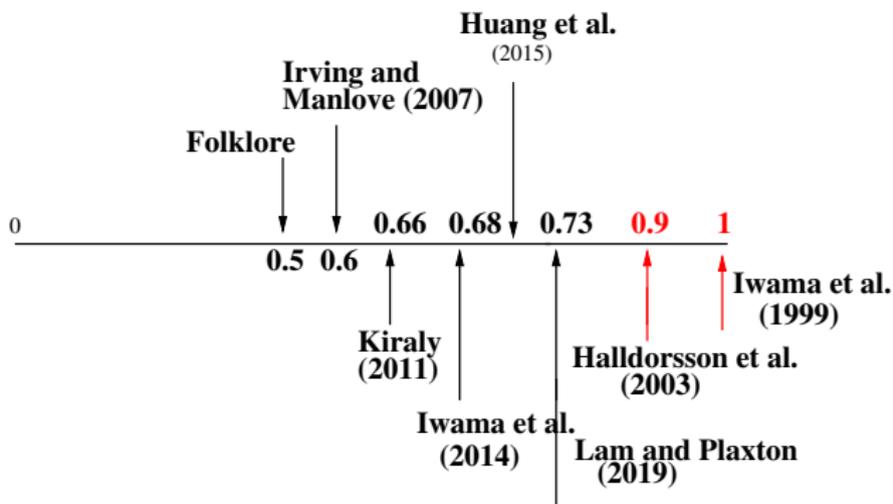
NP-hard even for restricted setting.

Naive method:

- Break ties arbitrarily.
- Run GS algo.

Variation #2: Incomplete Lists and Ties

Assume ties only on \mathcal{B} side.



Variation #2: Incomplete Lists and Ties

Király 2011

Assume ties only on \mathcal{B} side.

Király's Algorithm

- Break ties arbitrarily.
- Execute GS algo.
- Unmatched \mathcal{A} 's propose again with increased priority.
 - b uses increased priority for breaking ties.
 - Stability is not violated.

Variation #2: Incomplete Lists and Ties

Király 2011

a_1 : b_1 b_2
 a_2 : b_1

b_1 : (a_1 a_2)
 b_2 : a_1

Variation #2: Incomplete Lists and Ties

Király 2011

a_1 : b_1 b_2
 a_2 : b_1

b_1 : a_1 a_2
 b_2 : a_1

Király's Algo.

- Break ties arbitrarily.
- Run GS algo.
- Unmatched \mathcal{A} s propose with increased priority.

Variation #2: Incomplete Lists and Ties

Király 2011

a_1 : b_1 b_2
 a_2 : b_1

b_1 : a_1 a_2
 b_2 : a_1

Király's Algo.

- Break ties arbitrarily. ■ $a_1 \rightarrow b_1$
- Run GS algo.
- Unmatched \mathcal{A} s propose with increased priority.

Variation #2: Incomplete Lists and Ties

Király 2011

a_1 : $\boxed{b_1}$ b_2
 a_2 : b_1

b_1 : $\boxed{a_1}$ a_2
 b_2 : a_1

Király's Algo.

- Break ties arbitrarily.
 - Run GS algo.
 - Unmatched \mathcal{A} s propose with increased priority.
- $a_1 \rightarrow b_1$ accept.

Variation #2: Incomplete Lists and Ties

Király 2011

a_1 : $\boxed{b_1}$ b_2
 a_2 : b_1

b_1 : $\boxed{a_1}$ a_2
 b_2 : a_1

Király's Algo.

- Break ties arbitrarily.
- Run GS algo.
- Unmatched \mathcal{A} s propose with increased priority.
- $a_1 \rightarrow b_1$ accept.
- $a_2 \rightarrow b_1$

Variation #2: Incomplete Lists and Ties

Király 2011

a_1 : $\boxed{b_1}$ b_2
 a_2 : b_1

b_1 : $\boxed{a_1}$ a_2
 b_2 : a_1

Király's Algo.

- Break ties arbitrarily.
- Run GS algo.
- Unmatched \mathcal{A} s propose with increased priority.
- $a_1 \rightarrow b_1$ accept.
- $a_2 \rightarrow b_1$ reject.

Variation #2: Incomplete Lists and Ties

Király 2011

a_1 : $\boxed{b_1}$ b_2
 a_2 : b_1

b_1 : $\boxed{a_1}$ a_2
 b_2 : a_1

Király's Algo.

- Break ties arbitrarily.
 - Run GS algo.
 - Unmatched \mathcal{A} s propose with increased priority.
- $a_1 \rightarrow b_1$ accept.
 - $a_2 \rightarrow b_1$ reject.
 - $a_2^* \rightarrow b_1$ accept; recall ties originally.

Variation #2: Incomplete Lists and Ties

Király 2011

a_1 : b_1 b_2
 a_2 : $\boxed{b_1}$

b_1 : a_1 $\boxed{a_2}$
 b_2 : a_1

Király's Algo.

- Break ties arbitrarily.
 - Run GS algo.
 - Unmatched \mathcal{A} s propose with increased priority.
 - $a_1 \rightarrow b_1$ accept.
 - $a_2 \rightarrow b_1$ reject.
 - $a_2^* \rightarrow b_1$ accept; recall ties originally.
-

Variation #2: Incomplete Lists and Ties

Király 2011

a_1 : b_1 b_2
 a_2 : b_1

b_1 : a_1 a_2
 b_2 : a_1

Király's Algo.

- Break ties arbitrarily.
 - Run GS algo.
 - Unmatched \mathcal{A} s propose with increased priority.
 - $a_1 \rightarrow b_1$ accept.
 - $a_2 \rightarrow b_1$ reject.
 - $a_2^* \rightarrow b_1$ accept; recall ties originally.
 - $a_1 \rightarrow b_2$ accept.
-

Variation #2: Incomplete Lists and Ties

Király 2011

a_1 : b_1 b_2
 a_2 : b_1

b_1 : a_1 a_2
 b_2 : a_1

Király's Algo.

- Break ties arbitrarily.
- Run GS algo.
- Unmatched \mathcal{A} s propose with increased priority.
- $a_1 \rightarrow b_1$ accept.
- $a_2 \rightarrow b_1$ reject.
- $a_2^* \rightarrow b_1$ accept; recall ties originally.
- $a_1 \rightarrow b_2$ accept.

$$M = \{(a_1, b_2), (a_2, b_1)\}.$$

Variation #2: Incomplete Lists and Ties

Király 2011

$$a_1: \quad b_1 \quad \boxed{b_2}$$

$$a_2: \quad \boxed{b_1}$$

$$b_1: \quad a_1 \quad \boxed{a_2}$$

$$b_2: \quad \boxed{a_1}$$

Király's Algo.

- Break ties arbitrarily.
- Run GS algo.
- Unmatched A s propose with increased priority.
- $a_1 \rightarrow b_1$ accept.
- $a_2 \rightarrow b_1$ reject.
- $a_2^* \rightarrow b_1$ accept; recall ties originally.
- $a_1 \rightarrow b_2$ accept.

$$M = \{(a_1, b_2), (a_2, b_1)\}.$$

Goal: Argue about the size of the matching.

Variation #2: Incomplete Lists and Ties

Király 2011

$$a_1: \quad b_1 \quad \boxed{b_2}$$

$$a_2: \quad \boxed{b_1}$$

$$b_1: \quad a_1 \quad \boxed{a_2}$$

$$b_2: \quad \boxed{a_1}$$

Király's Algo.

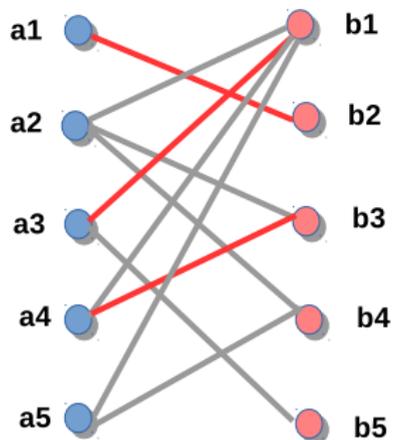
- Break ties arbitrarily.
- Run GS algo.
- Unmatched \mathcal{A} s propose with increased priority.
- $a_1 \rightarrow b_1$ accept.
- $a_2 \rightarrow b_1$ reject.
- $a_2^* \rightarrow b_1$ accept; recall ties originally.
- $a_1 \rightarrow b_2$ accept.

$$M = \{(a_1, b_2), (a_2, b_1)\}.$$

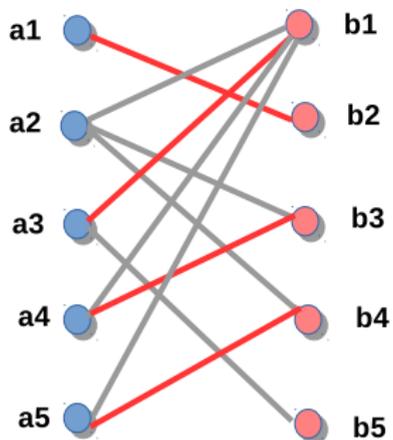
Goal: Argue about the size of the matching.

- Show no **short** aug. paths.

Matchings and aug. paths: a detour

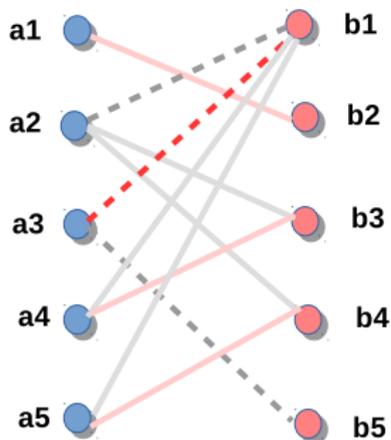


Matchings and aug. paths: a detour



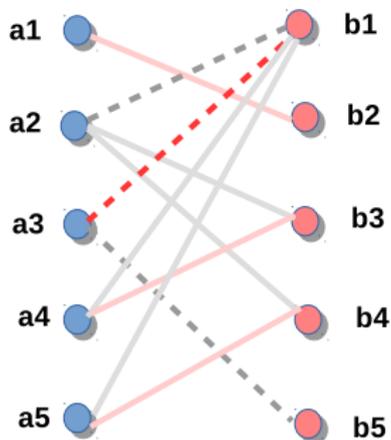
- Is this the largest sized matching?

Matchings and aug. paths: a detour



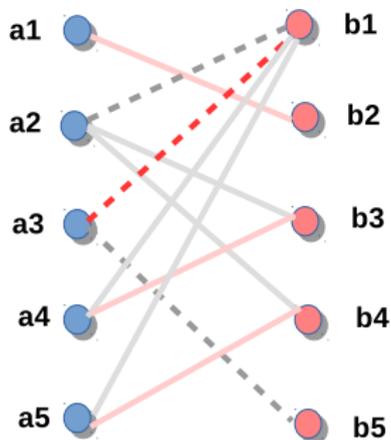
- Is this the largest sized matching?
- a_2, b_1, a_3, b_5 – alternating path with both end points free.

Matchings and aug. paths: a detour



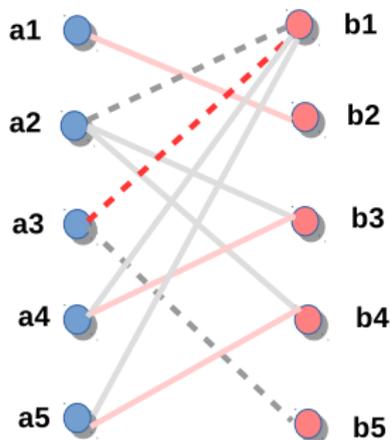
- Is this the largest sized matching?
- a_2, b_1, a_3, b_5 – alternating path with both end points free.
- Aug. paths: odd number of edges
(1, 3, 5, ..., $2k+1$)

Matchings and aug. paths: a detour



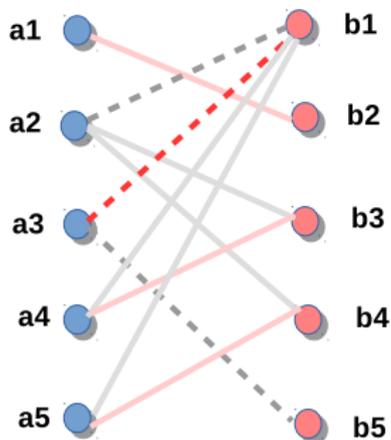
- Is this the largest sized matching?
- a_2, b_1, a_3, b_5 – alternating path with both end points free.
- Aug. paths: odd number of edges
(1, 3, 5, ..., $2k+1$)
- No one length aug. path \rightarrow maximal

Matchings and aug. paths: a detour



- Is this the largest sized matching?
 - a_2, b_1, a_3, b_5 – alternating path with both end points free.
 - Aug. paths: odd number of edges
(1, 3, 5, ..., $2k+1$)
 - No one length aug. path \rightarrow maximal
 - No **short** aug. path, closer to max. matching.
- Matching M without 1 and 3 length aug. paths.

Matchings and aug. paths: a detour



- Matching M without 1 and 3 length aug. paths.
- $|M| \geq \frac{2}{3}|M^*|$.

- Is this the largest sized matching?
- a_2, b_1, a_3, b_5 – alternating path with both end points free.
- Aug. paths: odd number of edges
(1, 3, 5, ..., $2k+1$)
- No one length aug. path \rightarrow maximal
- No **short** aug. path, closer to max. matching.

Matchings and aug. paths: a detour

M be any matching in a graph

M^* be a maximum cardinality matching.

Matchings and aug. paths: a detour

M be any matching in a graph

M^* be a maximum cardinality matching.

$$\text{Let } t = |M^*| - |M|$$

Matchings and aug. paths: a detour

M be any matching in a graph

M^* be a maximum cardinality matching.

$$\text{Let } t = |M^*| - |M|$$

Observe: M contains exactly t augmenting paths with respect to it.

Matchings and aug. paths: a detour

M be any matching in a graph

M^* be a maximum cardinality matching.

$$\text{Let } t = |M^*| - |M|$$

Observe: M contains exactly t augmenting paths with respect to it.

Assume: M does not have short augmenting paths. That is, the shortest augmenting path w.r.t. M is at least $2k + 1$.

Matchings and aug. paths: a detour

M be any matching in a graph

M^* be a maximum cardinality matching.

$$\text{Let } t = |M^*| - |M|$$

Observe: M contains exactly t augmenting paths with respect to it.

Assume: M does not have short augmenting paths. That is, the shortest augmenting path w.r.t. M is at least $2k + 1$.

$$t \leq \frac{|M|}{k}$$

Matchings and aug. paths: a detour

M be any matching in a graph

M^* be a maximum cardinality matching.

$$\text{Let } t = |M^*| - |M|$$

Observe: M contains exactly t augmenting paths with respect to it.

Assume: M does not have short augmenting paths. That is, the shortest augmenting path w.r.t. M is at least $2k + 1$.

$$t \leq \frac{|M|}{k} \quad \text{why?}$$

Matchings and aug. paths: a detour

M be any matching in a graph

M^* be a maximum cardinality matching.

$$\text{Let } t = |M^*| - |M|$$

Observe: M contains exactly t augmenting paths with respect to it.

Assume: M does not have short augmenting paths. That is, the shortest augmenting path w.r.t. M is at least $2k + 1$.

$$t \leq \frac{|M|}{k} \quad \text{why?}$$

$$|M^*| = |M| + t$$

Matchings and aug. paths: a detour

M be any matching in a graph

M^* be a maximum cardinality matching.

$$\text{Let } t = |M^*| - |M|$$

Observe: M contains exactly t augmenting paths with respect to it.

Assume: M does not have short augmenting paths. That is, the shortest augmenting path w.r.t. M is at least $2k + 1$.

$$t \leq \frac{|M|}{k} \quad \text{why?}$$

$$|M^*| = |M| + t \leq |M| + |M|/k$$

Matchings and aug. paths: a detour

M be **any matching** in a graph

M^* be a **maximum cardinality** matching.

$$\text{Let } t = |M^*| - |M|$$

Observe: M contains **exactly** t augmenting paths with respect to it.

Assume: M does not have **short** augmenting paths. That is, the shortest augmenting path w.r.t. M is at least $2k + 1$.

$$t \leq \frac{|M|}{k} \quad \text{why?}$$

$$|M^*| = |M| + t \leq |M| + |M|/k$$

$$|M^*| \leq \frac{k+1}{k} |M|$$

Back to Király's algorithm

Recall Király's algorithm

- Break ties arbitrarily.
- Execute GS algo.
- Unmatched \mathcal{A} 's propose again with increased priority.

Recall Király's algorithm

- Break ties arbitrarily.
- Execute GS algo.
- Unmatched \mathcal{A} 's propose again with increased priority.
 - b uses increased priority for breaking ties.

Recall Király's algorithm

- Break ties arbitrarily.
- Execute GS algo.
- Unmatched \mathcal{A} 's propose again with increased priority.
 - b uses increased priority for breaking ties.
 - Stability is not violated.

Recall Király's algorithm

- Break ties arbitrarily.
- Execute GS algo.
- Unmatched \mathcal{A} 's propose again with increased priority.
 - b uses increased priority for breaking ties.
 - Stability is not violated.
- ~~Need to argue about the size of the output.~~

Recall Király's algorithm

- Break ties arbitrarily.
- Execute GS algo.
- Unmatched \mathcal{A} 's propose again with increased priority.
 - b uses increased priority for breaking ties.
 - Stability is not violated.
- ~~■ Need to argue about the size of the output.~~
- Show that there are no short (1 and 3 length) aug. paths.

Recall Király's algorithm

- Break ties arbitrarily.
- Execute GS algo.
- Unmatched \mathcal{A} 's propose again with increased priority.
 - b uses increased priority for breaking ties.
 - Stability is not violated.
- ~~Need to argue about the size of the output.~~
- Show that there are no short (1 and 3 length) aug. paths.

Some observations:

Recall Király's algorithm

- Break ties arbitrarily.
- Execute GS algo.
- Unmatched \mathcal{A} 's propose again with increased priority.
 - b uses increased priority for breaking ties.
 - Stability is not violated.
- ~~■ Need to argue about the size of the output.~~
- Show that there are no short (1 and 3 length) aug. paths.

Some observations:

- If a woman b is unmatched at the end of algo., she never got a proposal.

Recall Király's algorithm

- Break ties arbitrarily.
- Execute GS algo.
- Unmatched \mathcal{A} 's propose again with increased priority.
 - b uses increased priority for breaking ties.
 - Stability is not violated.
- ~~Need to argue about the size of the output.~~
- Show that there are no short (1 and 3 length) aug. paths.

Some observations:

- If a woman b is unmatched at the end of algo., she never got a proposal.
- If a man a is unmatched at the end of algo., he got increased priority.

Output of Király's algorithm

Suppose there exists a 3 length aug. path w.r.t. M_{algo} .



Output of Király's algorithm

Suppose there exists a 3 length aug. path w.r.t. M_{algo} .



- a_2 never proposed to b_2 ($\because b_2$ is unmatched after algo)
→ a_2 did not get increased priority.

Output of Király's algorithm

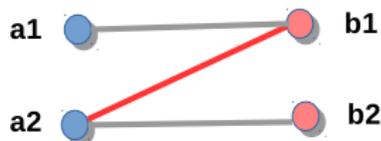
Suppose there exists a 3 length aug. path w.r.t. M_{algo} .



- a_2 never proposed to b_2 ($\because b_2$ is unmatched after algo)
 - $\rightarrow a_2$ did not get increased priority.
 - \rightarrow a_2 strictly prefers b_1 over b_2 .

Output of Király's algorithm

Suppose there exists a 3 length aug. path w.r.t. M_{algo} .



- a_2 never proposed to b_2 ($\because b_2$ is unmatched after algo)
 - $\rightarrow a_2$ did not get increased priority.
 - \rightarrow a_2 strictly prefers b_1 over b_2 .
- a_1 is unmatched at the end of algo.
 - $\rightarrow a_1$ must have got high priority.

Output of Király's algorithm

Suppose there exists a 3 length aug. path w.r.t. M_{algo} .



- a_2 never proposed to b_2 ($\because b_2$ is unmatched after algo)
 - $\rightarrow a_2$ did not get increased priority.
 - \rightarrow a_2 strictly prefers b_1 over b_2 .
- a_1 is unmatched at the end of algo.
 - $\rightarrow a_1$ must have got high priority.
 - \rightarrow b_1 strictly prefers a_2 over a_1 .

Output of Király's algorithm

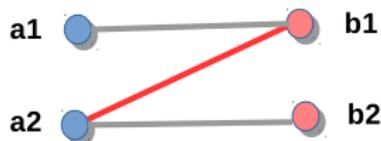
Suppose there exists a 3 length aug. path w.r.t. M_{algo} .



- a_2 never proposed to b_2 ($\because b_2$ is unmatched after algo)
 - $\rightarrow a_2$ did not get increased priority.
 - \rightarrow a_2 strictly prefers b_1 over b_2 .
- a_1 is unmatched at the end of algo.
 - $\rightarrow a_1$ must have got high priority.
 - \rightarrow b_1 strictly prefers a_2 over a_1 .
- (a_2, b_1) is a blocking pair w.r.t. M^* .

Output of Király's algorithm

Suppose there exists a 3 length aug. path w.r.t. M_{algo} .



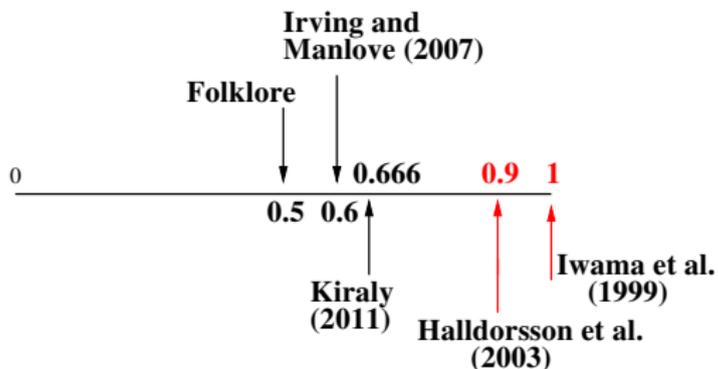
- a_2 never proposed to b_2 ($\because b_2$ is unmatched after algo)
 - $\rightarrow a_2$ did not get increased priority.
 - \rightarrow a_2 strictly prefers b_1 over b_2 .
- a_1 is unmatched at the end of algo.
 - $\rightarrow a_1$ must have got high priority.
 - \rightarrow b_1 strictly prefers a_2 over a_1 .
- (a_2, b_1) is a blocking pair w.r.t. M^* .
- contradicts stability of M^* .

There are no 3 length aug. paths w.r.t. M_{algo} .

Thus, $|M_{algo}| \geq \frac{2}{3}|M^*|$.

Variation #2: Incomplete Lists and Ties

Assume ties only on B side.



Main takeaways

- A simple extension of GS algo.
- A useful idea of **increased priority** that we will revisit.

Models : Recap

Model	Details	Goal	
Classical setting	strict and complete list	Compute a stable matching	✓
Variation #1	strict and incomplete list	Compute a larger optimal matching	
Variation #2	ties and incomplete list	Compute a largest stable matching	✓

Popularity – an alternative to stability

Gärdenfors 1975

Compare two matchings by **votes** of participants.

a_1 : b_1 b_2
 a_2 : b_1

b_1 : a_1 a_2
 b_2 : a_1

■ $M_s = \{(a_1, b_1)\}$

■ $M = \{(a_2, b_1)\}$

	M_s	M
a_1	✓	
a_2		✓
b_1	✓	

- M_s beats M w.r.t. popularity.
- **Popular Matching:**
One which cannot be beaten!
- **Q:** Does a popular matching exist?

Popularity – an alternative to stability

Gärdenfors 1975

Compare two matchings by **votes** of participants.

a_1 : $\boxed{b_1}$ $\boxed{b_2}$
 a_2 : $\boxed{b_1}$

b_1 : $\boxed{a_1}$ $\boxed{a_2}$
 b_2 : $\boxed{a_1}$

■ $M_s = \{(a_1, b_1)\}$

■ $M' = \{(a_1, b_2), (a_2, b_1)\}$

	M_s	M
a_1	✓	
a_2		✓
b_1	✓	
b_2		✓

- Neither one beats each other.
- Does **not prove** popularity.
- **Q**: Does a popular matching exist?
Yes! A stable matching is popular.

Variation #1: Incomplete Lists

- Preferences are **strict** and can be **incomplete**.

a_1 : b_1 b_2
 a_2 : b_1

b_1 : a_1 a_2
 b_2 : a_1

- Does a stable matching exist? **Yes!** $M = \{(a_1, b_1)\}$.

Question: Are there larger optimal matchings?

Yes! $M' = \{(a_1, b_2), (a_2, b_1)\}$ is **popular**.

Variation #1: Incomplete Lists

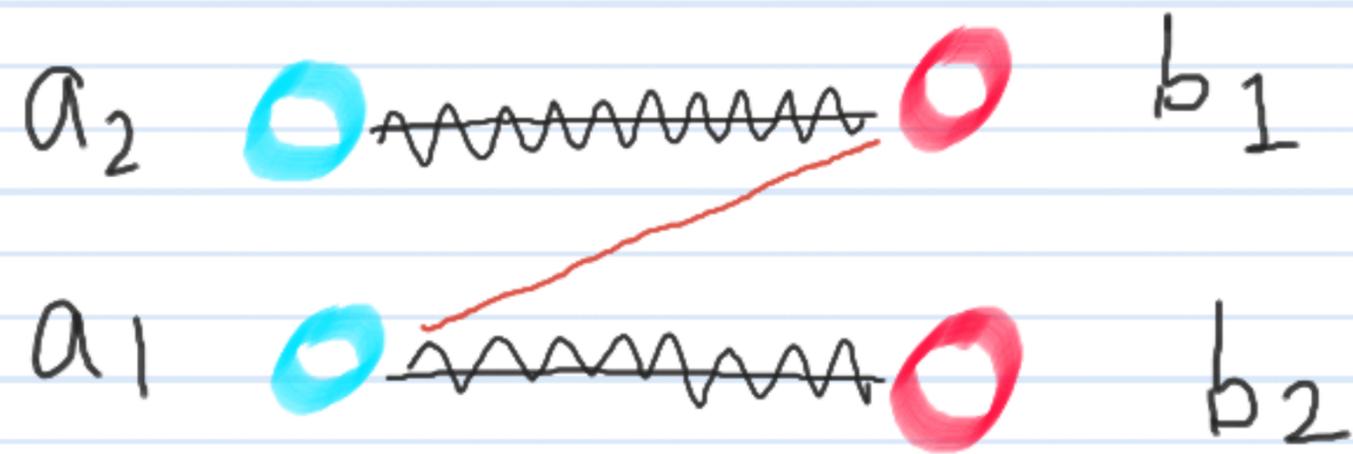
Kavitha 2012

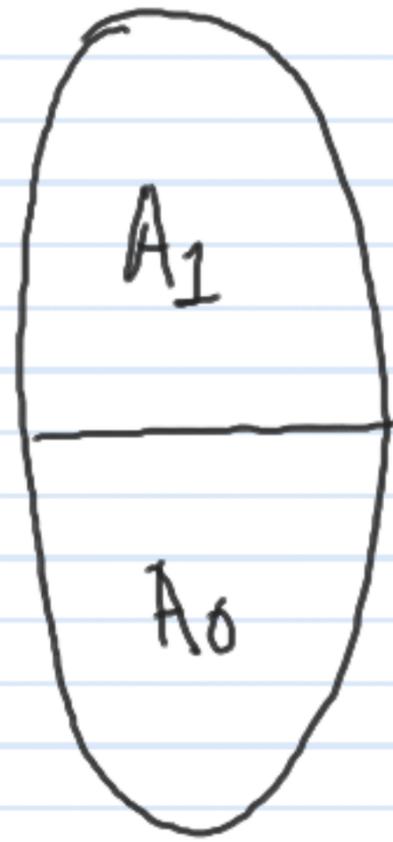
$$\begin{array}{l} a_1: \quad b_1 \quad b_2 \\ a_2: \quad b_1 \end{array}$$

$$\begin{array}{l} b_1: \quad a_1 \quad a_2 \\ b_2: \quad a_1 \end{array}$$

Goal: Compute Largest sized Popular matching.

- Run GS algo. Unmatched \mathcal{A} 's propose with increased priority.
- Stability may be violated.
- Guarantees on output :
 - M_{algo} is max. sized **popular**.
 - $|M_{algo}| \geq |M_s|$ and $|M_{algo}| \geq \frac{2}{3}|M^*|$.
 - Linear time algo.



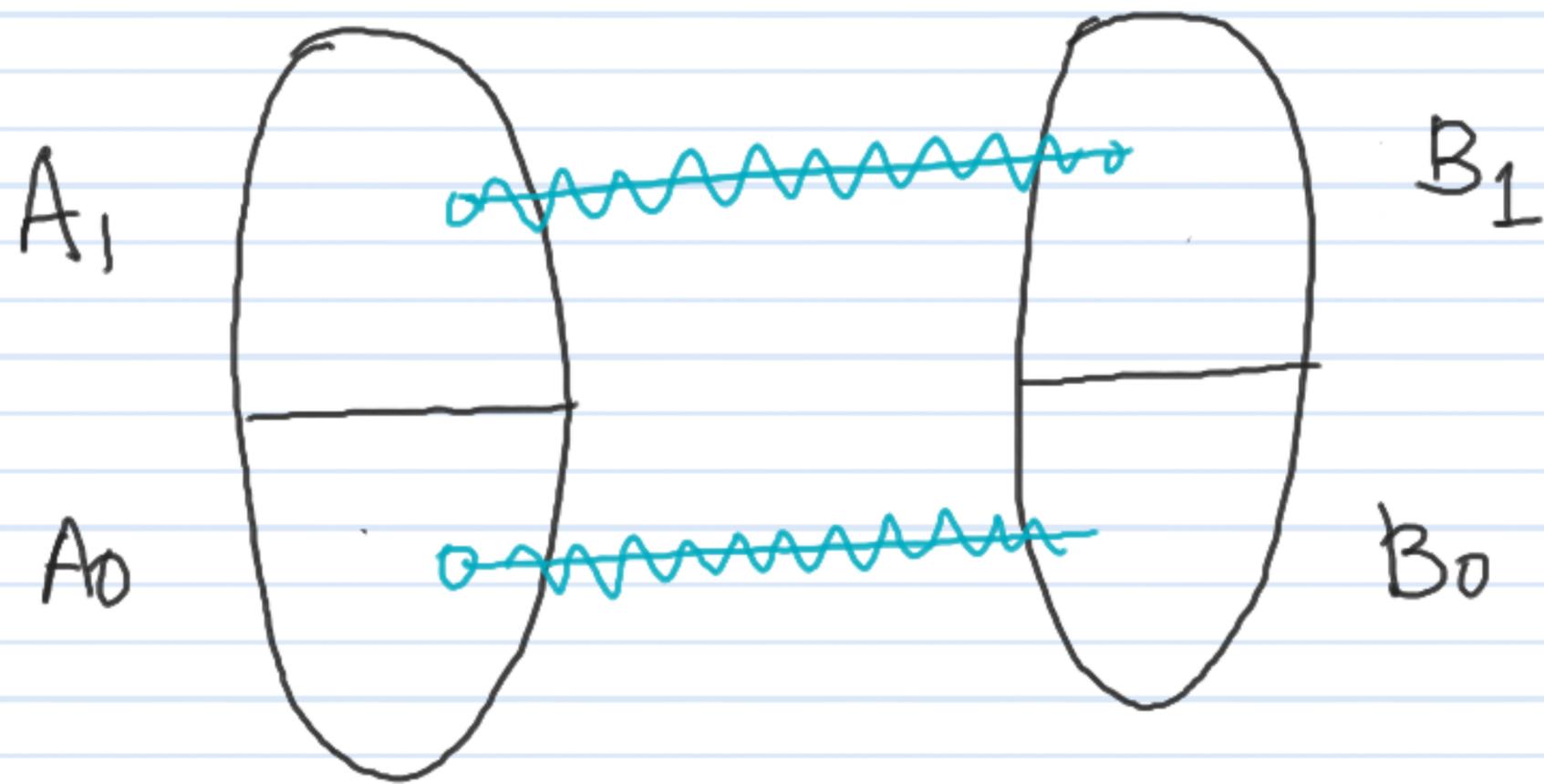


A_1 : set of men who got \neq status

$$A_0 = A \setminus A_1$$

B_1 : set of women matched to men in A_1

$$B_0 = B \setminus B_1$$



① Matched edges are horizontal

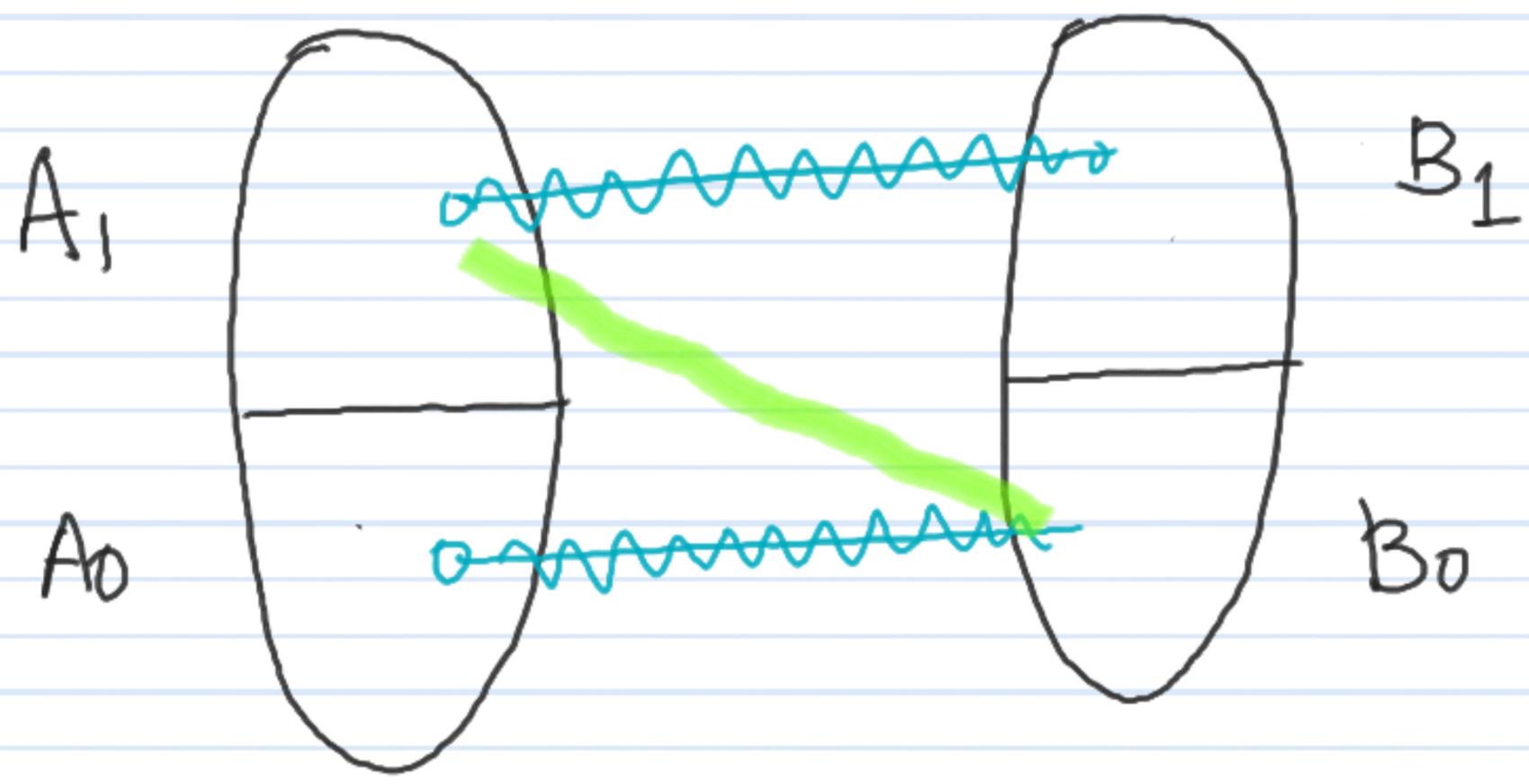
② Unmatched edges can be anywhere.

A_1 : set of men who got \neq status

$$A_0 = A \setminus A_1$$

B_1 : set of women matched to men in A_1

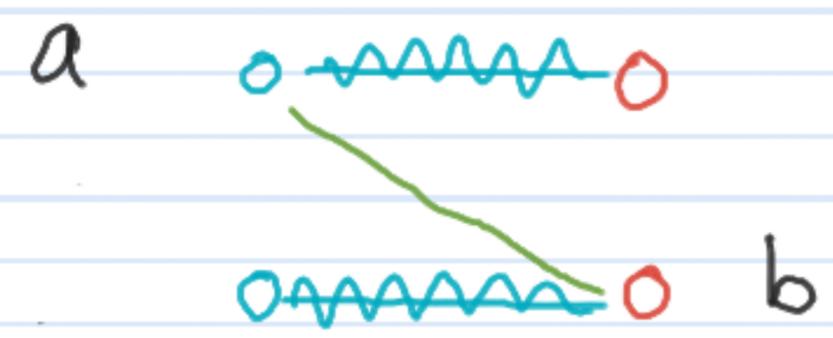
$$B_0 = B \setminus B_1$$



- ① Matched edges are horizontal
- ② Unmatched edges can be anywhere.

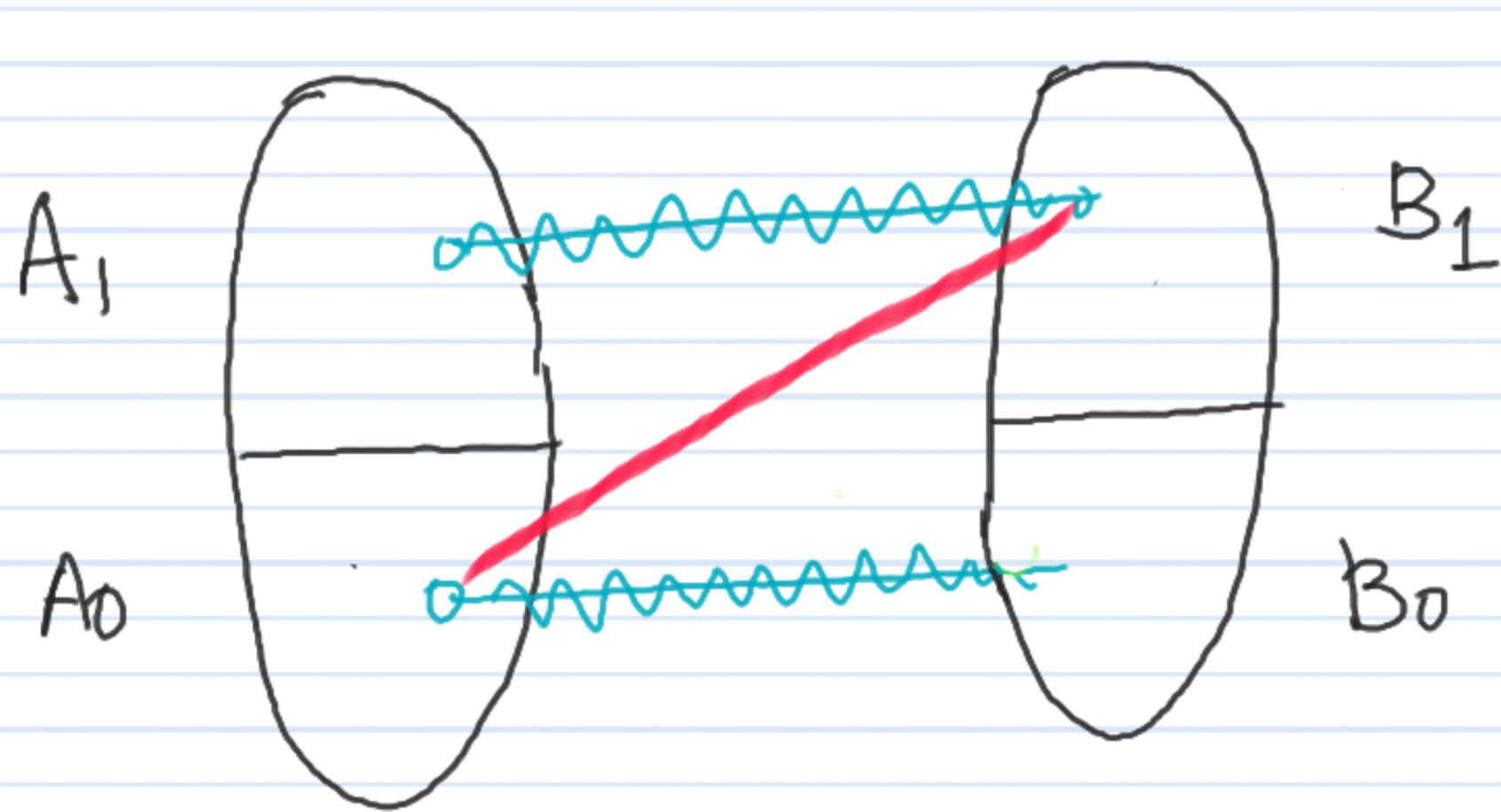
$A_1 B_0$ edges:

claim:



- (1) a prefers b over $M(a)$
- (2) b prefers a over $M(b)$

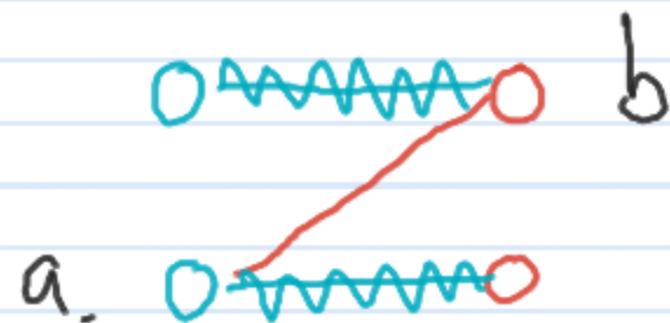
} why??



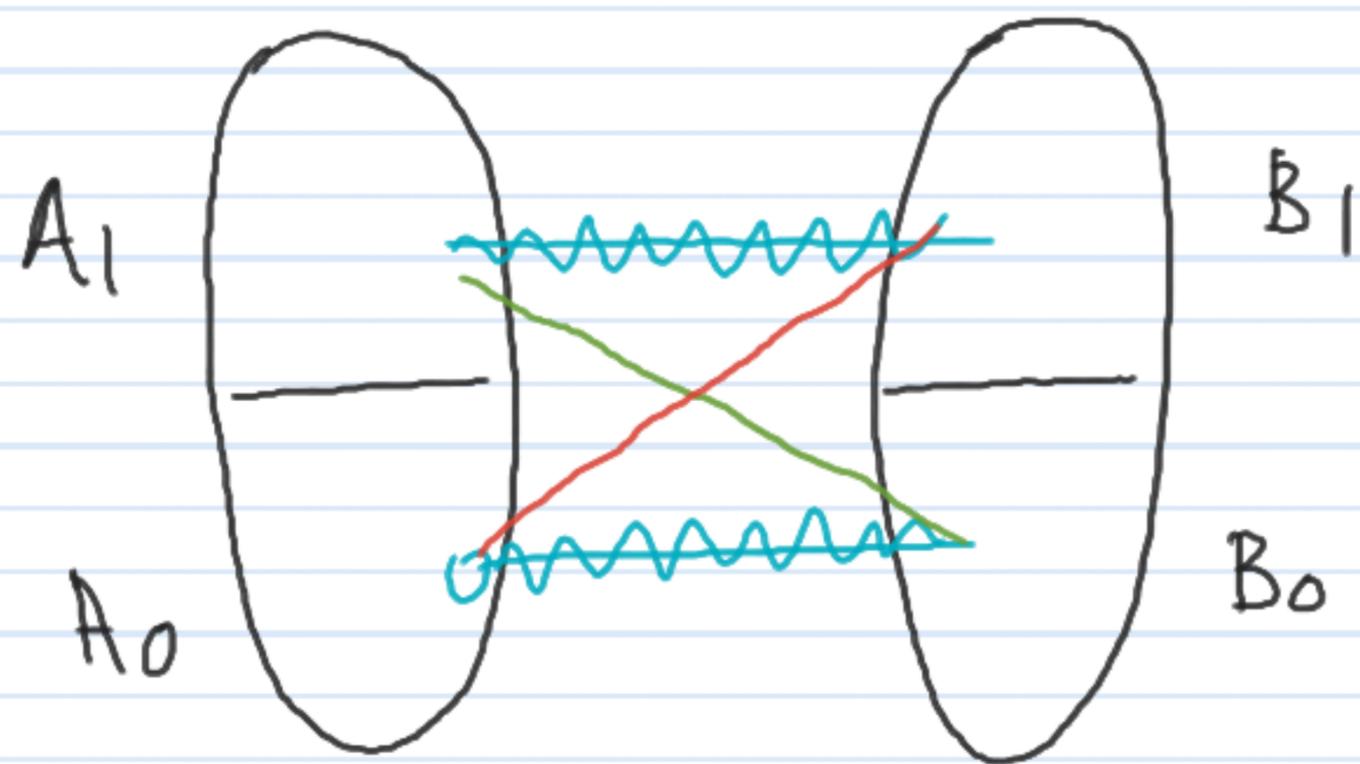
① Matched edges are horizontal

② Unmatched edges can be anywhere.

$A_0 B_1$ edges :



It can happen that both a and b prefer each other to their current partners in M



green edges : good for M (give us 2 votes)

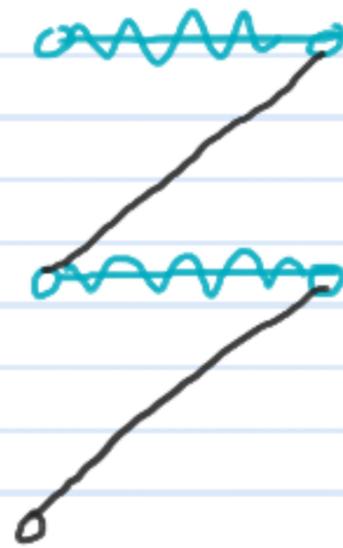
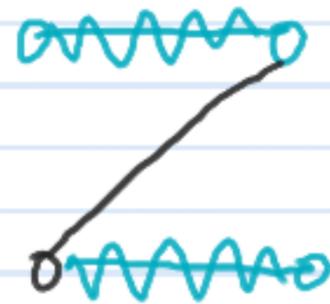
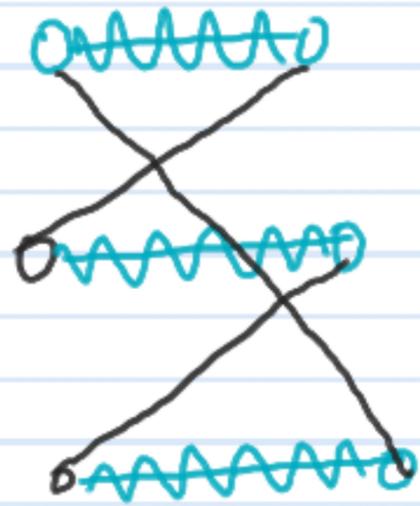
red edges : bad for M (M loses 2 votes)

other edges : horizontal unmatched edges?

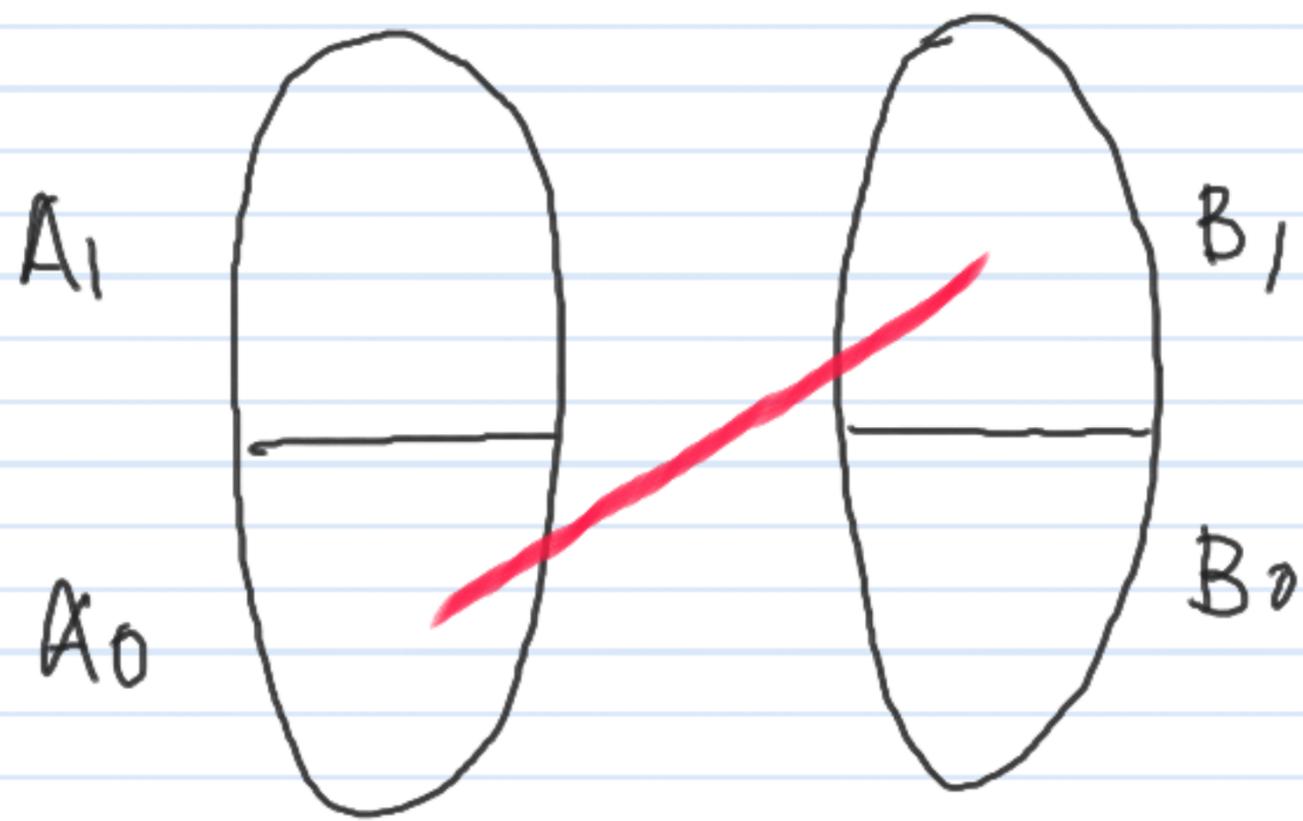
Claim: M produced by algo is popular.

$\Rightarrow \nexists N$ s.t. N gets more votes over M .

Symmetric difference of M and N



If N has to get more votes, it gets more votes along one of the components.



If cycle lies completely in $A_1 B_1$ or $A_0 B_0$ it cannot get more votes.

Cycle must take lots of $A_0 B_1$ edges

Can a cycle take lots of $A_0 B_1$ edges

No! For each $A_0 B_1$ edge it must have at least one $B_0 A_1$ edge which is a good edge.

Models : Summary

Model	Details	Goal	
Classical setting	strict and complete list	Compute a stable matching	✓
Variation #1	strict and incomplete list	Compute a larger optimal matching	✓
Variation #2	ties and incomplete list	Compute a largest stable matching	✓

To summarize..

A simple extension of GS algo.

- Each case requires different proof techniques and several non-trivial details.
- All algorithms can be written as a **reduction** to a suitable SM instance.
- Works in the presence of capacities (upper quotas).

To summarize..

A simple extension of GS algo.

- Each case requires different proof techniques and several non-trivial details.
- All algorithms can be written as a **reduction** to a suitable SM instance.
- Works in the presence of capacities (upper quotas).

To summarize..

A simple extension of GS algo.

- Each case requires different proof techniques and several non-trivial details.
- All algorithms can be written as a **reduction** to a suitable SM instance.
- Works in the presence of capacities (upper quotas).

Thank You!