Popular Matchings

Kavitha Telikepalli (TIFR, Mumbai)

[Joint work with Chien-Chung Huang]

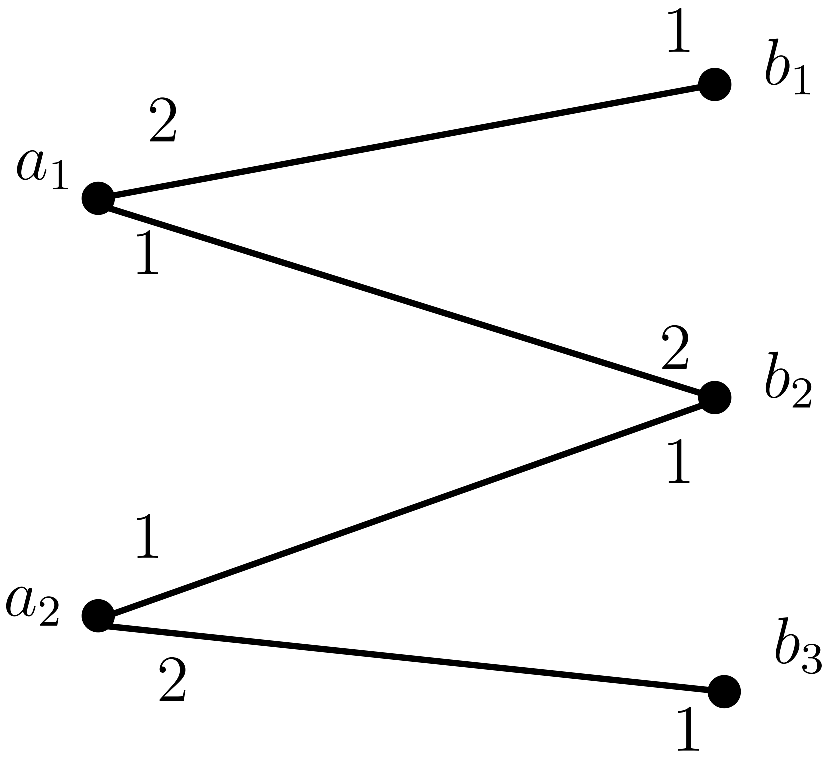
3rd Annual Mysore Park Theory Workshop: August 2012

The input graph

- Input: a bipartite graph $G = (\mathcal{A} \cup \mathcal{B}, E)$.

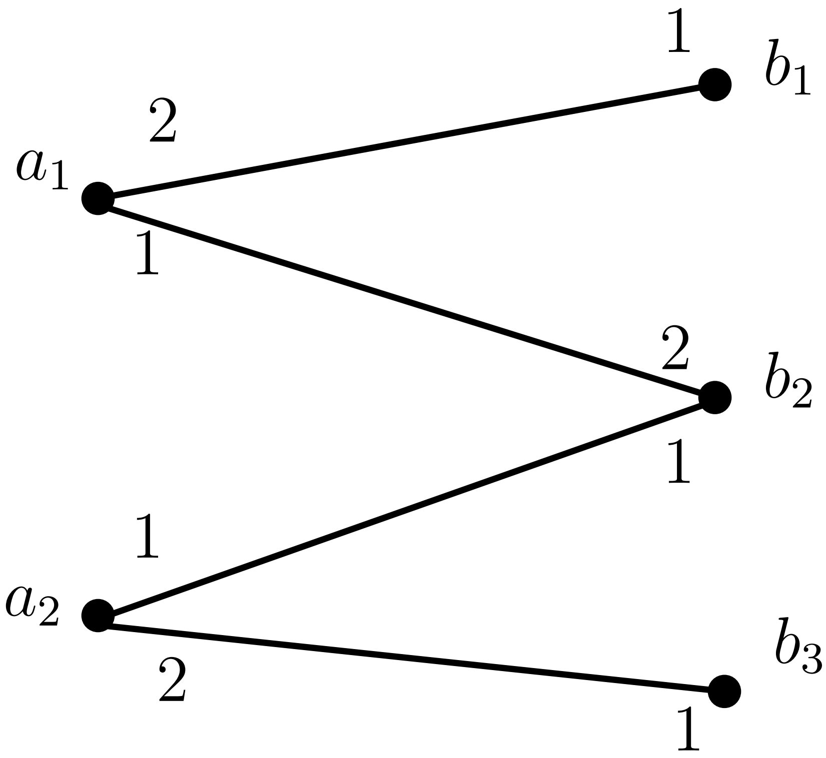
The input graph

- Input: a bipartite graph $G = (\mathcal{A} \cup \mathcal{B}, E)$.



The input graph

- Input: a bipartite graph $G = (\mathcal{A} \cup \mathcal{B}, E)$.



- $\mathcal{A}$: set of students; $\mathcal{B}$: set of advisers.

The input

- Each $u \in \mathcal{A} \cup \mathcal{B}$ ranks its neighbors in a strict order of preference.

The input

- Each $u \in \mathcal{A} \cup \mathcal{B}$ ranks its neighbors in a strict order of preference.
- Problem: compute a "good" matching in G.

The input

- Each $u \in \mathcal{A} \cup \mathcal{B}$ ranks its neighbors in a strict order of preference.
- Problem: compute a "good" matching in G.
  - every vertex is *selfish*

The input

- Each $u \in \mathcal{A} \cup \mathcal{B}$ ranks its neighbors in a strict order of preference.
- Problem: compute a “good” matching in G.
  - every vertex is *selfish*
    - u wants to be matched to its best ranked neighbor who is willing to be matched to u.

Optimal matchings

- Let M be the matching obtained.

Optimal matchings

- Let M be the matching obtained.

  - The following property should hold for every u:

Optimal matchings

- Let M be the matching obtained.
  - The following property should hold for every u:
    - there is no neighbor ranked better than $M(u)$ who is willing to be matched to u.

Optimal matchings

- Let M be the matching obtained.
  - The following property should hold for every u:
    - there is no neighbor ranked better than $M(u)$ who is willing to be matched to u.
  - Such a matching M is *stable*.

Stable Matchings

- A matching M is stable if it has no “blocking edges”.

Stable Matchings

- A matching M is stable if it has no "blocking edges".
  - edge (u, v) blocks M if u and v prefer each other to their respective assignments in M.

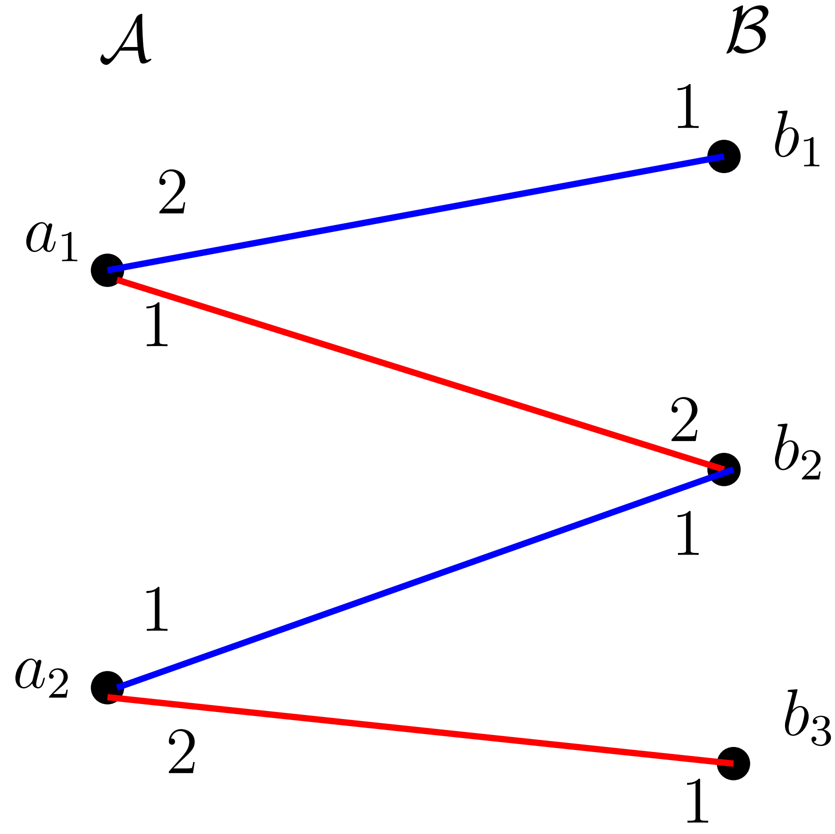
Stable Matchings

- A matching M is stable if it has no "blocking edges".
  - edge (u, v) blocks M if u and v prefer each other to their respective assignments in M.
    - u is unmatched or prefers v to $M(u)$

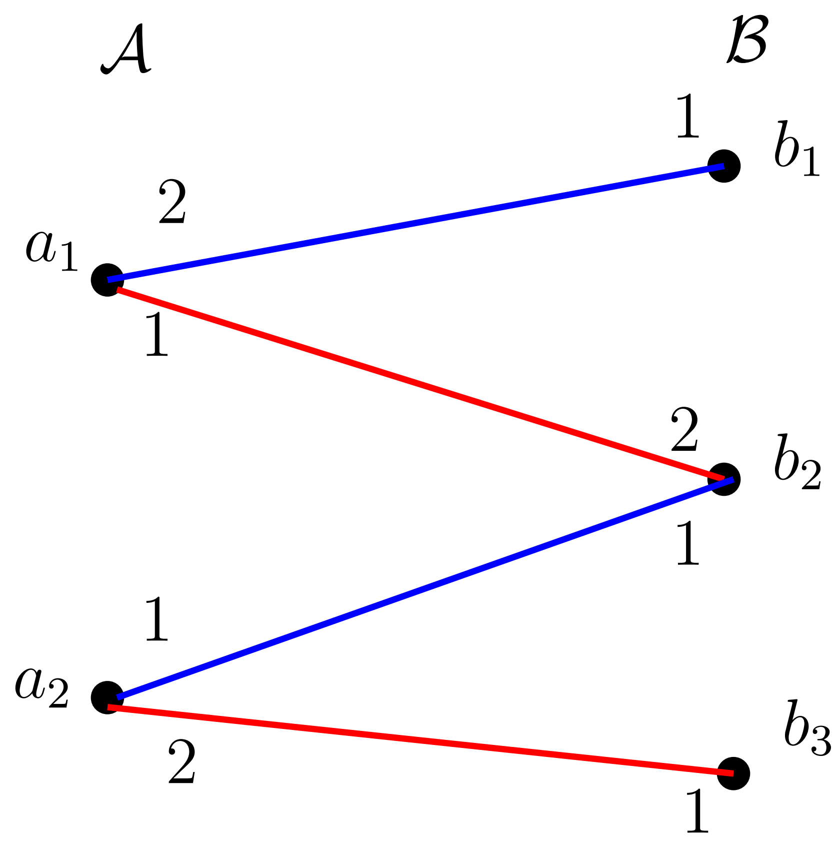
Stable Matchings

- A matching M is stable if it has no “blocking edges”.
  - edge (u, v) blocks M if u and v prefer each other to their respective assignments in M.
    - u is unmatched or prefers v to $M(u)$
    - v is unmatched or prefers u to $M(v)$.

Stable matchings



Stable matchings



- The blue matching is stable while the red is not.

Stable matchings

- Do stable matchings always exist?

Stable matchings

- Do stable matchings always exist?
  - Yes; also such a matching can be computed in linear time [Gale-Shapley, 62].

Stable matchings

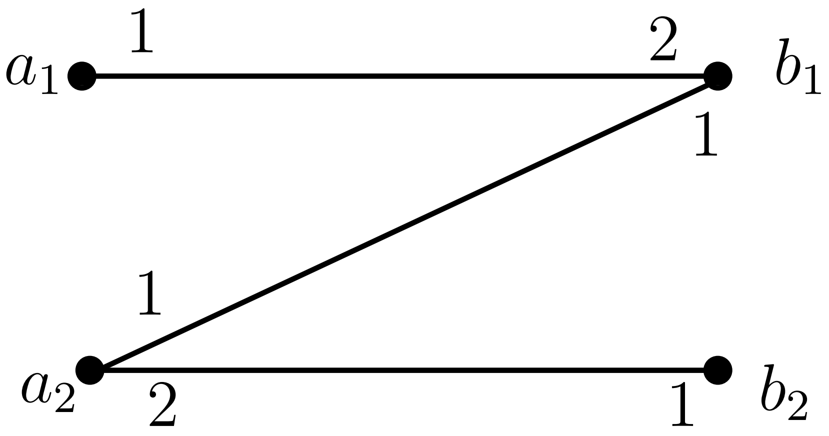
- Do stable matchings always exist?
  - Yes; also such a matching can be computed in linear time [Gale-Shapley, 62].
- *Gale-Shapley algorithm*: Men (vertices of A) propose and Women (those in B) dispose.

Gale-Shapley algorithm for stable matchings

- *Men (vertices of $\mathcal{A}$) propose and Women (those in $\mathcal{B}$) dispose.*

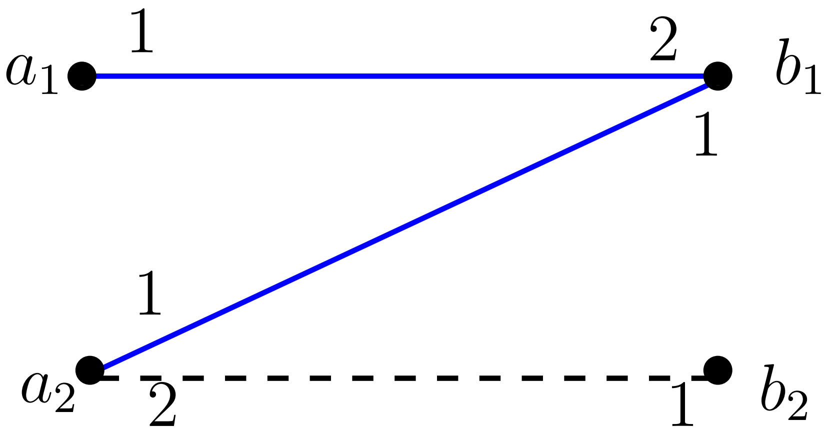
Gale-Shapley algorithm for stable matchings

- *Men (vertices of $\mathcal{A}$) propose and Women (those in $\mathcal{B}$) dispose.*



Gale-Shapley algorithm for stable matchings

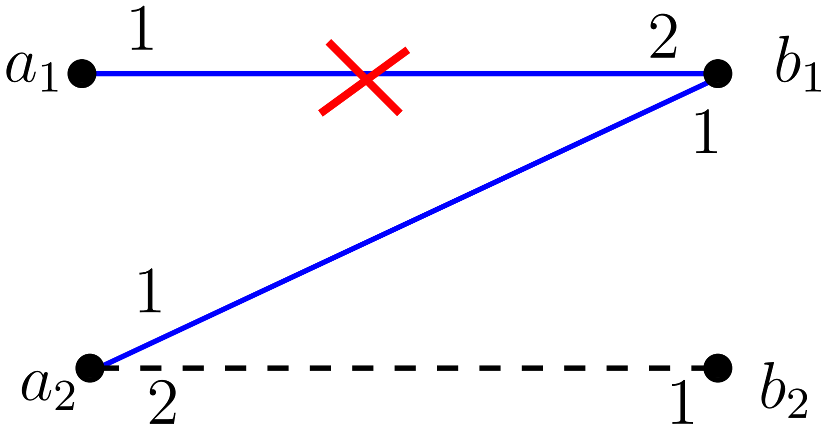
- *Men (vertices of $\mathcal{A}$) propose and Women (those in $\mathcal{B}$) dispose.*



- a_1 proposes to his top neighbor b_1; so does a_2.

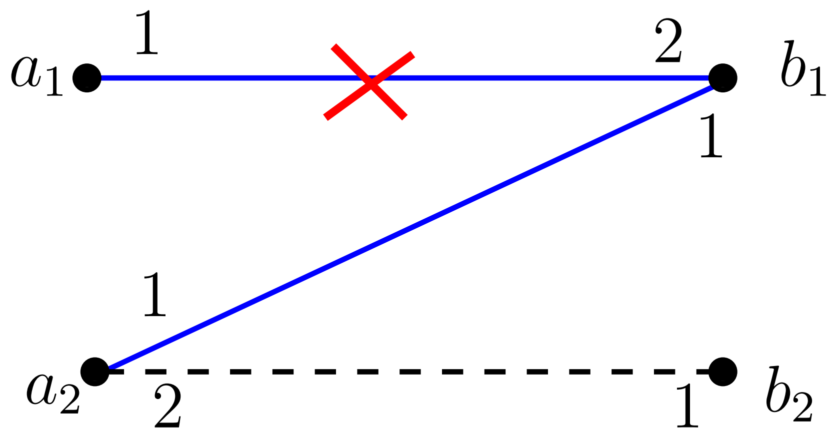
Gale-Shapley algorithm for stable matchings

- b_1 rejects a_1 and accepts a_2.



Gale-Shapley algorithm for stable matchings

- b_1 rejects a_1 and accepts a_2.



- The algorithm terminates when every man is either rejected by all his nbrs or gets matched to some nbr.

Price of stability

- Ideally, M_{max} is the optimal matching.

Price of stability

- Ideally, M_{max} is the optimal matching.

- Size of a stable matching:

Price of stability

- Ideally, M_{max} is the optimal matching.
- Size of a stable matching:
  - all stable matchings in G have the same size.

Price of stability

- Ideally, M_{max} is the optimal matching.
- Size of a stable matching:
  - all stable matchings in G have the same size.
  - |stable matching| could be as low as $|M_{max}|/2$.

Popular matchings

- A new notion of optimality that is a compromise between M_{max} and a stable matching?

Popular matchings

- A new notion of optimality that is a compromise between M_{max} and a stable matching?

- A notion based on *popularity*:

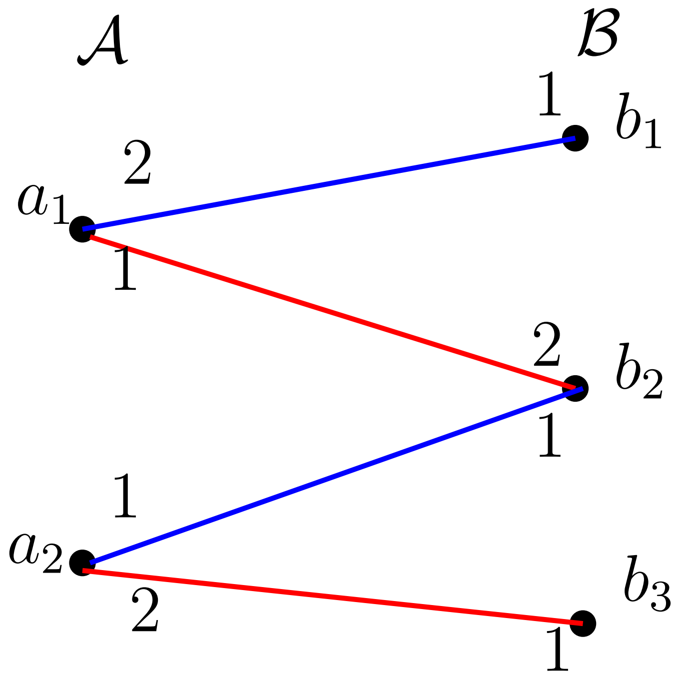
Popular matchings

- A new notion of optimality that is a compromise between M_{max} and a stable matching?

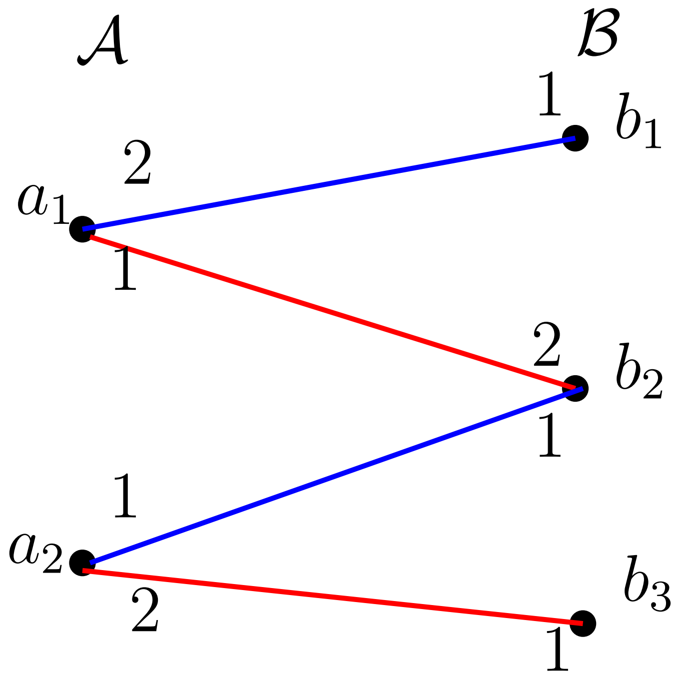
- A notion based on *popularity*:

Matching M_1 is *more popular* than matching M_2 if $|\{$vertices that prefer $M_1\}| > |\{$vertices that prefer $M_2\}|$.

An example

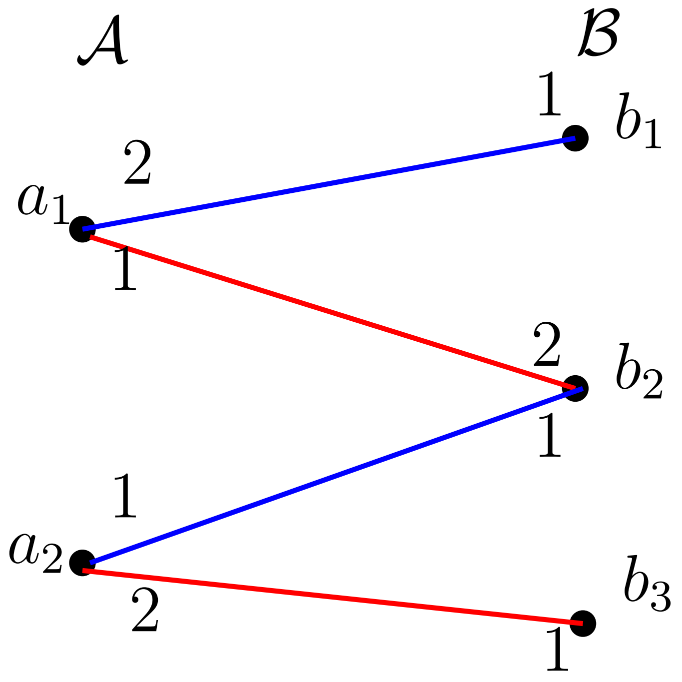


An example



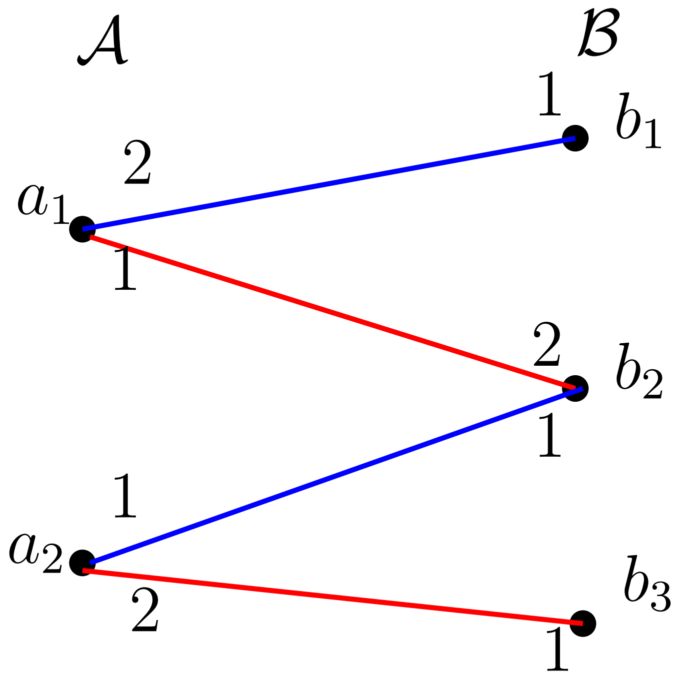
- a_1 and b_3 prefer the red matching

An example



- a_1 and b_3 prefer the red matching
- b_1, b_2, and a_2 prefer the blue matching

An example



- a_1 and b_3 prefer the red matching
- b_1, b_2, and a_2 prefer the blue matching
- blue matching is more popular than red matching.

Popular matchings

- We use $M_1 \succ M_2$ to denote the relation that M_1 is more popular than M_2.

Popular matchings

- We use $M_1 \succ M_2$ to denote the relation that M_1 is more popular than M_2.
- The "*more popular than*" relation is not transitive: we can have $M_1 \succ M_2 \succ M_3 \succ M_1$.

Popular matchings

- We use $M_1 \succ M_2$ to denote the relation that M_1 is more popular than M_2.
- The "*more popular than*" relation is not transitive: we can have $M_1 \succ M_2 \succ M_3 \succ M_1$.
- M is popular if there is *no* M' such that $M' \succ M$.

Popular matchings

M is popular $\Rightarrow$ for every matching M' we have:

$|\{\text{vertices that prefer } M'\}| \leq |\{\text{vertices that prefer } M\}|$.

Popular matchings

M is popular $\Rightarrow$ for every matching M' we have:

$|\{$vertices that prefer $M'\}| \leq |\{$vertices that prefer $M\}|$.

- Do popular matchings always exist in G?

Popular matchings

M is popular $\Rightarrow$ for every matching M' we have:

$|\{\text{vertices that prefer } M'\}| \leq |\{\text{vertices that prefer } M\}|$.

- Do popular matchings always exist in G?
    - yes, in fact, every stable matching is popular.

stable $\implies$ popular

- Comparing a stable matching S with any matching M:

stable $\Longrightarrow$ popular

- Comparing a stable matching S with any matching M:

$$u \text{ prefers } M \text{ to } S \Rightarrow M(u) \text{ has to prefer } S \text{ to } M.$$

stable $\Longrightarrow$ popular

- Comparing a stable matching S with any matching M:

  u prefers M to $S \Rightarrow M(u)$ has to prefer S to M.

- So number of votes for $M \leq$ number of votes for S.

stable $\Longrightarrow$ popular

- Comparing a stable matching S with any matching M:

  u prefers M to $S \Rightarrow M(u)$ has to prefer S to M.

- So number of votes for $M \leq$ number of votes for S.
  - So a stable matching is always popular.

stable $\Longrightarrow$ popular

- Comparing a stable matching S with any matching M:

  u prefers M to $S \Rightarrow M(u)$ has to prefer S to M.

- So number of votes for $M \leq$ number of votes for S.
  - So a stable matching is always popular.
    - In fact, it is a *minimum* size popular matching.

Stable matchings

- Let S be a stable matching and let M be a smaller matching.

Stable matchings

- Let S be a stable matching and let M be a smaller matching.
  - we will show that M has to be unpopular.

Stable matchings

- Let S be a stable matching and let M be a smaller matching.
  - we will show that M has to be unpopular.
- $|M| < |S|$, so $M \oplus S$ has an augmenting path p wrt M.

Stable matchings

- Let S be a stable matching and let M be a smaller matching.
  - we will show that M has to be unpopular.
- $|M| < |S|$, so $M \oplus S$ has an augmenting path p wrt M.
- *Claim:* $M \oplus p \succ M$.

Stable matchings

- Let S be a stable matching and let M be a smaller matching.
  - we will show that M has to be unpopular.
- $|M| < |S|$, so $M \oplus S$ has an augmenting path p wrt M.
- *Claim:* $M \oplus p \succ M$.
- $(M \oplus p)(u) = S(u)$ if $u \in p$,
  $(M \oplus p)(u) = M(u)$ otherwise.

The alternating path p

The alternating path p

- red: edges of M; black: edges of S.

The alternating path p

- red: edges of M; black: edges of S.
- both the endpoints of p prefer S to M.

The alternating path p

- red: edges of M; black: edges of S.
- both the endpoints of p prefer S to M.
- for every M-edge (u, v) in p:
  u prefers M to $S \Rightarrow v$ prefers S to M.
  (otherwise (u, v) would block S)

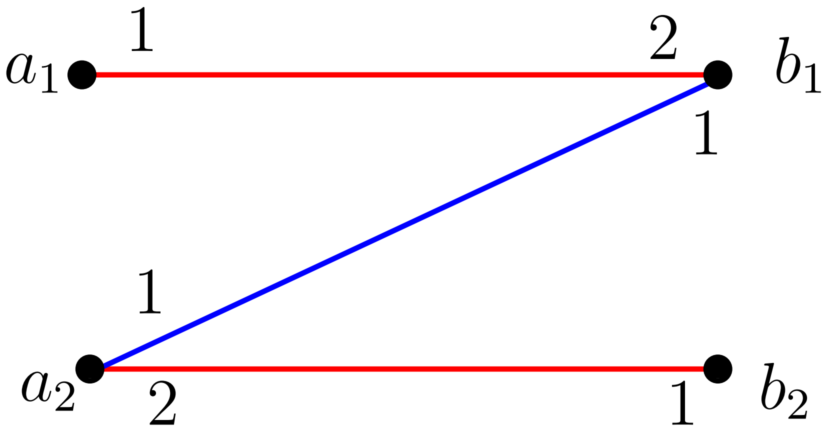
The alternating path p

- red: edges of M; black: edges of S.
- both the endpoints of p prefer S to M.
- for every M-edge (u, v) in p:
  u prefers M to $S \Rightarrow v$ prefers S to M.
  (otherwise (u, v) would block S)

- Thus restricted to p, we have $S \succ M$.
  So $M \oplus p \succ M$.

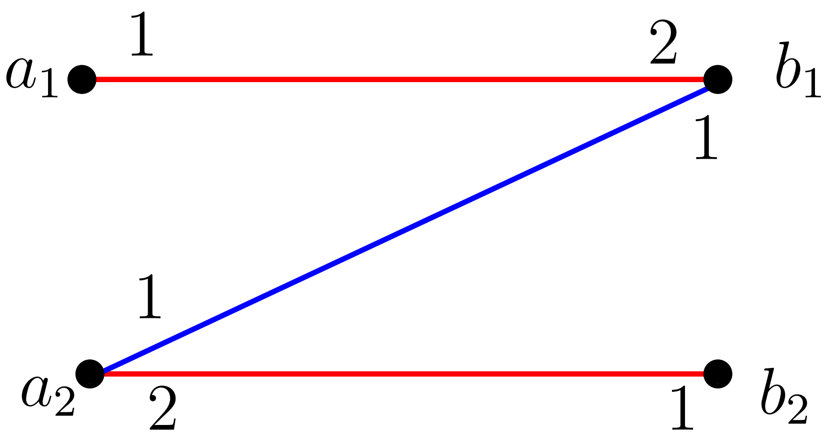
Min vs max size popular matchings

- The blue matching is a minimum size popular matching.



Min vs max size popular matchings

- The blue matching is a minimum size popular matching.



- The red matching is a maximum size popular matching.

Some questions

- Is |max size popular matching|$/|M_{max}|$ always larger than $1/2$?

Some questions

- Is |max size popular matching|$/|M_{max}|$ always larger than $1/2$?

- Can a maximum size popular matching be efficiently computed?

Some questions

- Is |max size popular matching|$/|M_{max}|$ always larger than $1/2$?
- Can a maximum size popular matching be efficiently computed?
- Characterization of a maximum size popular matching?

A characterization of popular matchings

- Call an edge (u, v) *negative* wrt M if

A characterization of popular matchings

- Call an edge (u, v) *negative* wrt M if

  u prefers $M(u)$ to v *and* v prefers $M(v)$ to u.

A characterization of popular matchings

- Call an edge (u, v) *negative* wrt M if

  u prefers $M(u)$ to v ***and*** v prefers $M(v)$ to u.

- Delete from G all negative edges wrt M — call this subgraph G_M.

A characterization of popular matchings

- M is popular $\iff$ M has these 3 properties in G_M:

A characterization of popular matchings

- M is popular $\iff$ M has these 3 properties in G_M:
  - no alternating cycle has a blocking edge.

A characterization of popular matchings

- M is popular $\iff$ M has these 3 properties in G_M:

  - no alternating cycle has a blocking edge.

  - no alternating path with a free endpoint has a blocking edge.

A characterization of popular matchings

- M is popular $\iff$ M has these 3 properties in G_M:
  - no alternating cycle has a blocking edge.
  - no alternating path with a free endpoint has a blocking edge.
  - no alternating path has 2 blocking edges.

Max size popular matchings

- In addition, if M has this 4th property:

Max size popular matchings

- In addition, if M has this 4th property:
  - there is no *augmenting path* wrt M in G_M.

Max size popular matchings

- In addition, if M has this 4th property:

  - there is no *augmenting path* wrt M in G_M.

$\Rightarrow$ any larger matching has to be *unpopular*.

Max size popular matchings

- In addition, if M has this 4th property:
  - there is no *augmenting path* wrt M in G_M.

$\Rightarrow$ any larger matching has to be *unpopular*.

  - That is, M will be a maximum size popular matching.

A first attempt

- Goal: To compute a matching that satisfies those 4 properties.

A first attempt

- Goal: To compute a matching that satisfies those 4 properties.

  - *Idea*: come up with a suitable partition (L, R) of $\mathcal{A} \cup \mathcal{B}$ such that

A first attempt

- Goal: To compute a matching that satisfies those 4 properties.
  - *Idea*: come up with a suitable partition (L, R) of $\mathcal{A} \cup \mathcal{B}$ such that
    - Gale-Shapley algorithm on (L, R) yields such a matching.

Our first algorithm (Huang and K., 2011)

- Run Gale-Shapley algorithm on $(\mathcal{A}, \mathcal{B})$: let S be this matching.

Our first algorithm (Huang and K., 2011)

- Run Gale-Shapley algorithm on $(\mathcal{A}, \mathcal{B})$: let S be this matching.
  - Set $L_1 =$ set of vertices left unmatched in S and $R_1 = (\mathcal{A} \cup \mathcal{B}) \setminus L_1$.

Our first algorithm (Huang and K., 2011)

- Run Gale-Shapley algorithm on $(\mathcal{A}, \mathcal{B})$: let S be this matching.
  - Set $L_1 =$ set of vertices left unmatched in S and $R_1 = (\mathcal{A} \cup \mathcal{B}) \setminus L_1$.
  - Run Gale-Shapley algorithm on (L_1, R_1): let M_1 be this matching.

Our first algorithm (Huang and K., 2011)

- Run Gale-Shapley algorithm on $(\mathcal{A}, \mathcal{B})$: let S be this matching.
  - Set $L_1 =$ set of vertices left unmatched in S and $R_1 = (\mathcal{A} \cup \mathcal{B}) \setminus L_1$.
  - Run Gale-Shapley algorithm on (L_1, R_1): let M_1 be this matching.
    - If M_1 is R_1-perfect, then M_1 satisfies those 4 properties.

Our first algorithm

- Else let A_1 be the set of unmatched men in R_1.

Our first algorithm

- Else let A_1 be the set of unmatched men in R_1.
  - Set $L_1' = L_1 \cup A_1$ and $R_1' = (\mathcal{A} \cup \mathcal{B}) \setminus L_1'$.

Our first algorithm

- Else let A_1 be the set of unmatched men in R_1.
  - Set $L_1' = L_1 \cup A_1$ and $R_1' = (\mathcal{A} \cup \mathcal{B}) \setminus L_1'$.
  - Run Gale-Shapley algorithm on (L_1', R_1'): let M_1' be this matching.

Our first algorithm

- Else let A_1 be the set of unmatched men in R_1.
  - Set $L_1' = L_1 \cup A_1$ and $R_1' = (\mathcal{A} \cup \mathcal{B}) \setminus L_1'$.
  - Run Gale-Shapley algorithm on (L_1', R_1'): let M_1' be this matching.
    - If M_1' is R_1'-perfect, then M_1' satisfies those 4 properties.

Our first algorithm

- Else let B_1 be the set of unmatched vertices in R'_1.

Our first algorithm

- Else let B_1 be the set of unmatched vertices in R_1'.
  - Note that $B_1 \subset \mathcal{B}$.

Our first algorithm

- Else let B_1 be the set of unmatched vertices in R'_1.
    - Note that $B_1 \subset \mathcal{B}$.
- Set $L_2 = L_1 \cup B_1$ and $R_2 = (\mathcal{A} \cup \mathcal{B}) \setminus L_2$.

Our first algorithm

- Else let B_1 be the set of unmatched vertices in R'_1.
  - Note that $B_1 \subset \mathcal{B}$.
- Set $L_2 = L_1 \cup B_1$ and $R_2 = (\mathcal{A} \cup \mathcal{B}) \setminus L_2$.
- Run Gale-Shapley algorithm on (L_2, R_2): let M_2 be this matching.

Our first algorithm

- If M_2 is R_2-perfect, then done.

Our first algorithm

- If M_2 is R_2-perfect, then done.

- Else move unmatched men from right to left and compute M_2'.

Our first algorithm

- If M_2 is R_2-perfect, then done.
- Else move unmatched men from right to left and compute M_2'.
  - if M_2' is R_2'-perfect, then done

Our first algorithm

- If M_2 is R_2-perfect, then done.
- Else move unmatched men from right to left and compute M_2'.
  - if M_2' is R_2'-perfect, then done
    - else move the above men back to the right

Our first algorithm

- If M_2 is R_2-perfect, then done.
- Else move unmatched men from right to left and compute M_2'.
  - if M_2' is R_2'-perfect, then done
    - else move the above men back to the right
    - move unmatched women from right to left

Our first algorithm

- If M_2 is R_2-perfect, then done.
- Else move unmatched men from right to left and compute M_2'.
  - if M_2' is R_2'-perfect, then done
    - else move the above men back to the right
    - move unmatched women from right to left
    - start the next round.

Our first algorithm

- The number of rounds is at most $|\mathcal{B}|$:

Our first algorithm

- The number of rounds is at most $|\mathcal{B}|$:
  - either round i is the last round or $L_{i+1} = L_i +$ at least 1 woman

Our first algorithm

- The number of rounds is at most $|\mathcal{B}|$:
  - either round i is the last round or $L_{i+1} = L_i +$ at least 1 woman
  - once a woman moves from right to left, she never moves back to the right side again.

Our first algorithm

- The number of rounds is at most $|\mathcal{B}|$:
  - either round i is the last round or $L_{i+1} = L_i +$ at least 1 woman
  - once a woman moves from right to left, she never moves back to the right side again.
- Running time: $O(m|\mathcal{B}|)$, where $m = |E|$.

Max size popular matching

- Result: an $O(mn_0)$ time algorithm to compute a max size popular matching. $(m = |E|,\ n_0 = \min(|\mathcal{A}|, |\mathcal{B}|))$.

Max size popular matching

- Result: an $O(mn_0)$ time algorithm to compute a max size popular matching. $(m = |E|,\ n_0 = \min(|\mathcal{A}|, |\mathcal{B}|))$.

- However a stable matching is faster to compute.

Max size popular matching

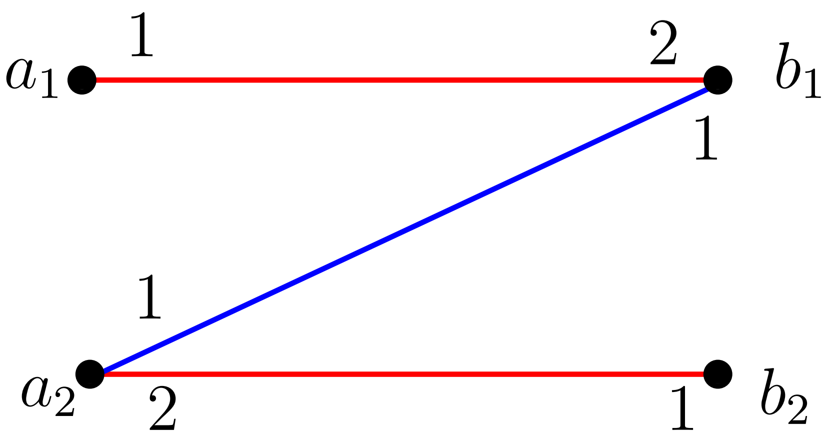
- Result: an $O(mn_0)$ time algorithm to compute a max size popular matching. $(m = |E|,\ n_0 = \min(|\mathcal{A}|, |\mathcal{B}|))$.

- However a stable matching is faster to compute.

- A linear time algorithm for maximum size popular matching?

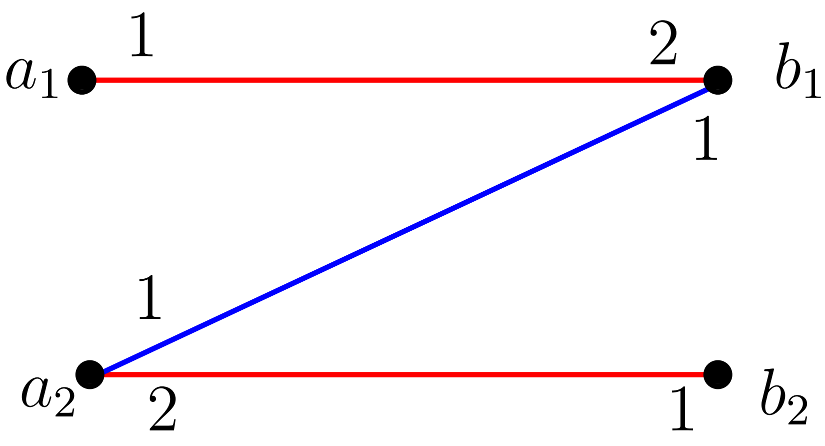
Stable vs max size popular matching

- The blue matching is stable.



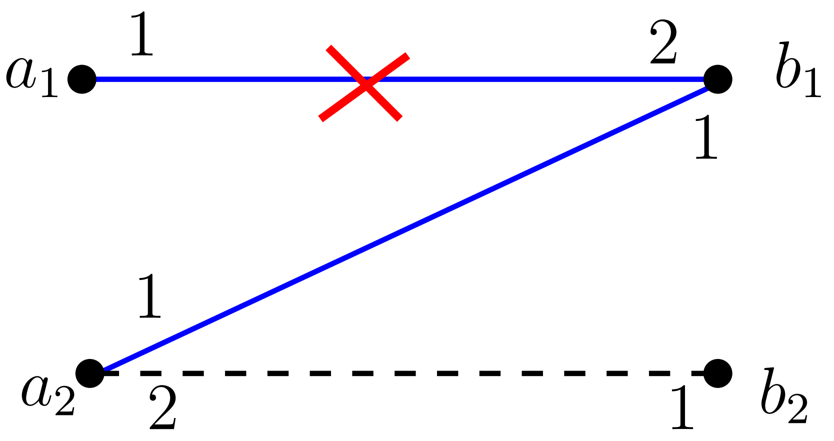
Stable vs max size popular matching

- The blue matching is stable.

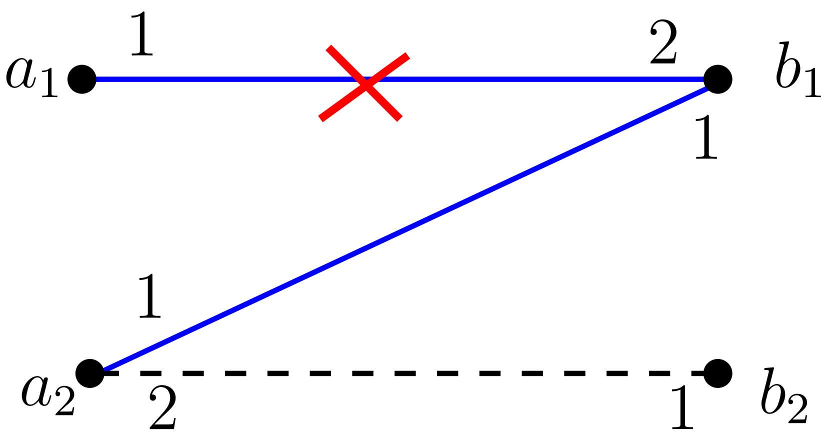


- The red matching is a maximum size popular matching.

Modifying Gale-Shapley ...

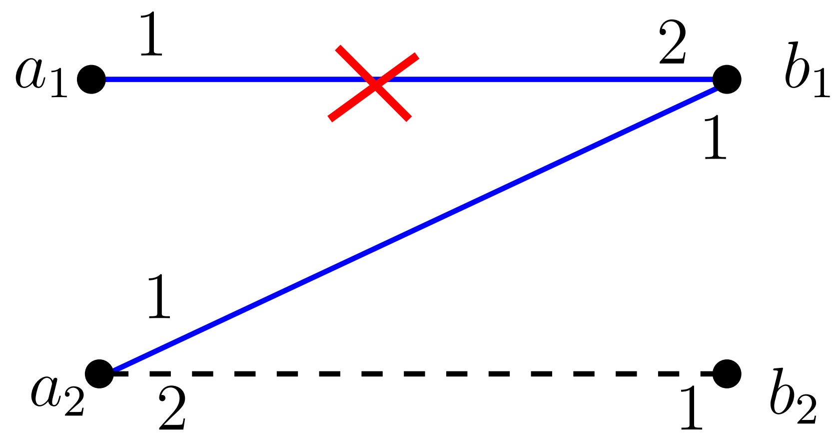


Modifying Gale-Shapley ...



- Modify the Gale-Shapley algorithm so that a_1 gets a "second chance" to propose to b_1.

Modifying Gale-Shapley ...



- Modify the Gale-Shapley algorithm so that a_1 gets a "second chance" to propose to b_1.
- when a_1 proposes for the *second* time to b_1, then b_1 should prefer a_1 to a_2.

Implementing this idea

- Have *two* copies a^0 and a^1 of every man a:

Implementing this idea

- Have *two* copies a^0 and a^1 of every man a:
  - there will be two edges (a^1, b) and (a^0, b) corresponding to every edge (a, b) in G.

Implementing this idea

- Have *two* copies a^0 and a^1 of every man a:
  - there will be two edges (a^1, b) and (a^0, b) corresponding to every edge (a, b) in G.
    - every woman prefers a level 1 nbr to a level 0 nbr.

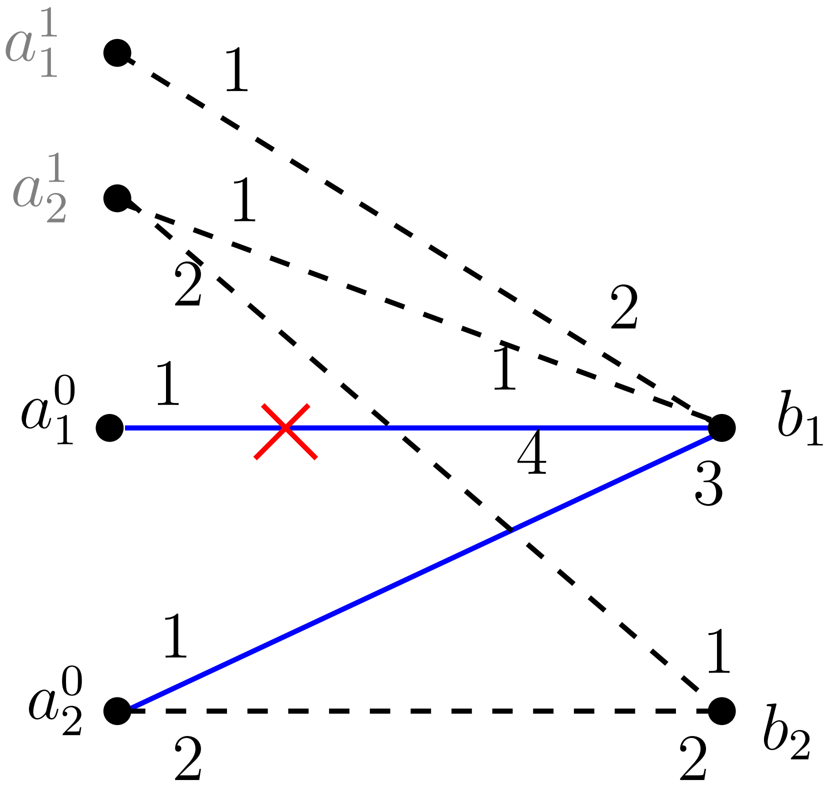
Implementing this idea

- Have *two* copies a^0 and a^1 of every man a:
  - there will be two edges (a^1, b) and (a^0, b) corresponding to every edge (a, b) in G.
    - every woman prefers a level 1 nbr to a level 0 nbr.
    - among level 1 nbrs: her original preference order.

Implementing this idea

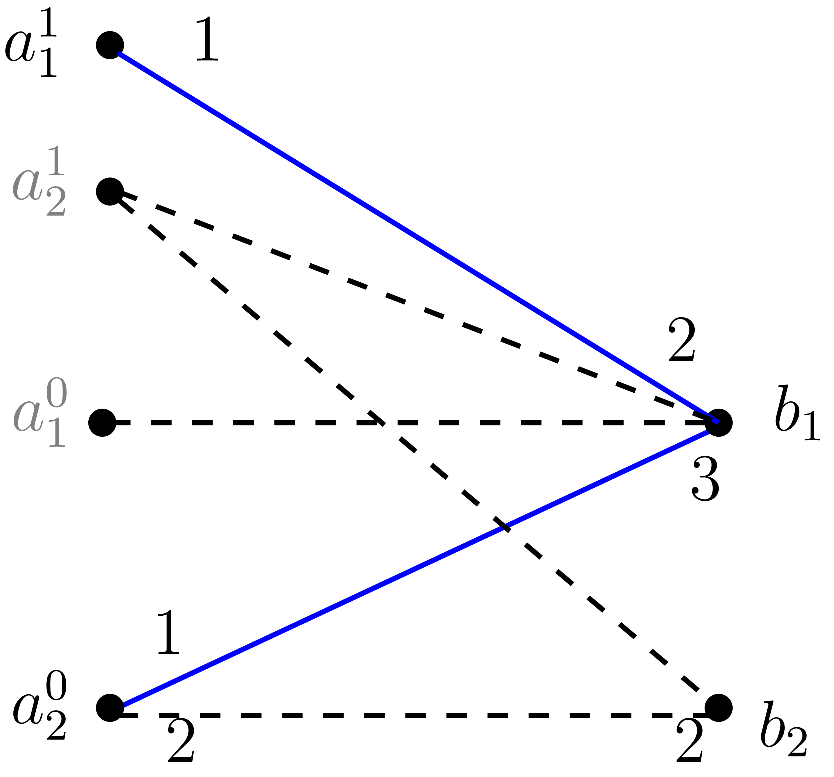
- Have *two* copies a^0 and a^1 of every man a:
  - there will be two edges (a^1, b) and (a^0, b) corresponding to every edge (a, b) in G.
    - every woman prefers a level 1 nbr to a level 0 nbr.
    - among level 1 nbrs: her original preference order.
    - among level 0 nbrs: her original preference order.

In the new graph



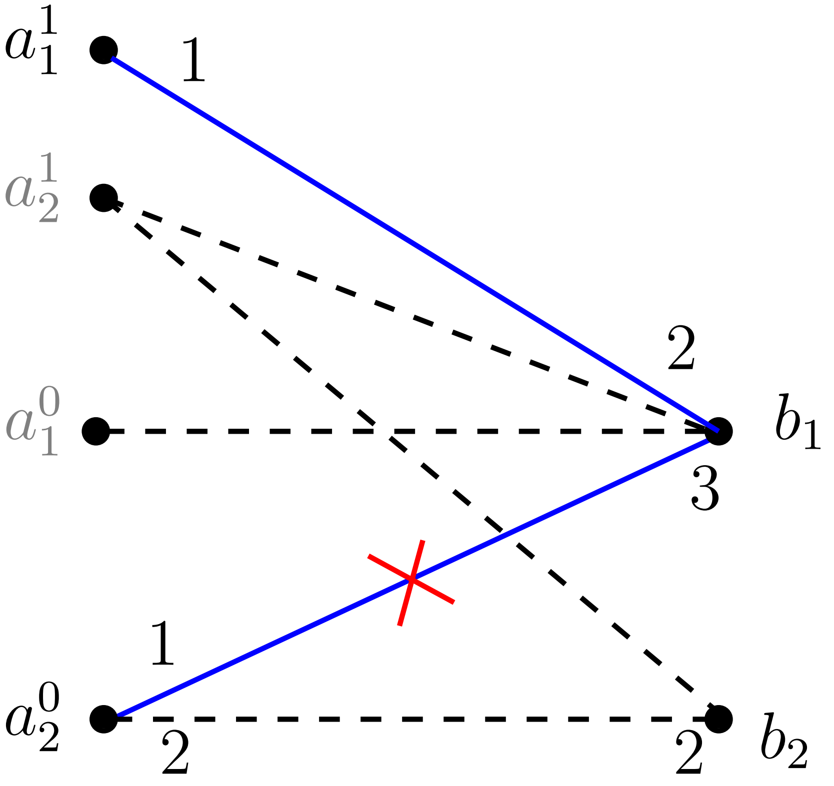
- a_1^0 is rejected by his only neighbor b_1.

In the new graph



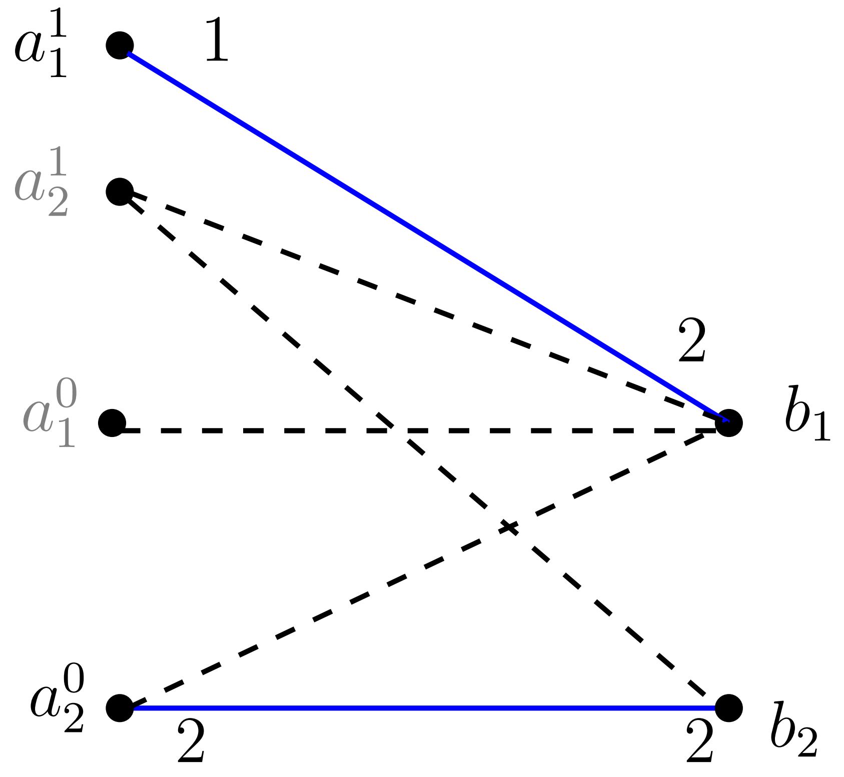
- So a_1^1 becomes active and proposes to b_1.

In the new graph



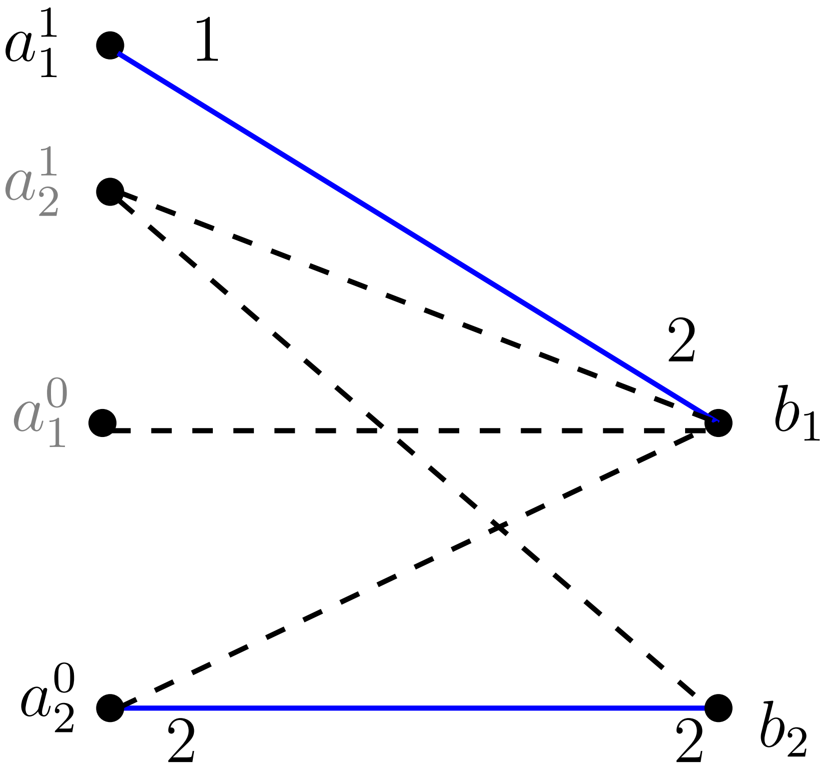
- b_1 accepts a_1^1 and rejects a_2^0.

In the new graph



- So a_2^0 proposes to his next preferred neighbor b_2.

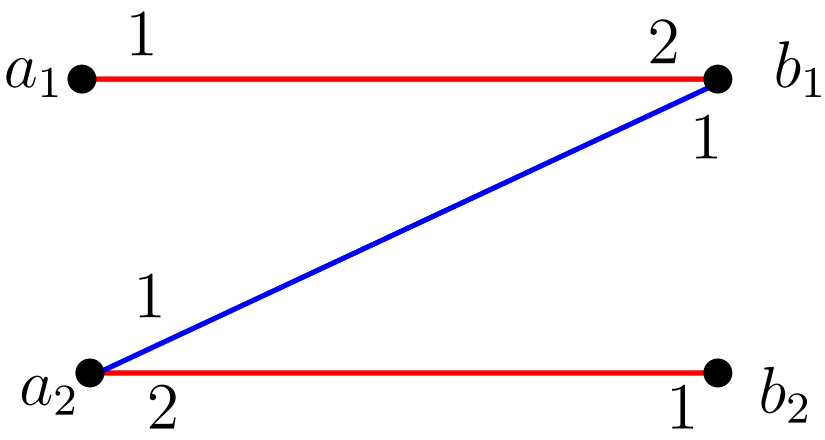
In the new graph



- The matching $\{(a_1^1, b_1),\ (a_2^0, b_2)\}$ is computed.

Back in the original graph

- Thus OPT $= \{(a_1, b_1), (a_2, b_2)\}$, the red matching, is found.



A linear time algorithm (K., 2012)

- Let $\tilde{G}_2$ be the graph on $\tilde{A}_2 \cup \mathcal{B}$ where $\tilde{A}_2$ consists of two copies a^0 and a^1 of each $a \in \mathcal{A}$.

A linear time algorithm (K., 2012)

- Let $\tilde{G}_2$ be the graph on $\tilde{A}_2 \cup \mathcal{B}$ where $\tilde{A}_2$ consists of two copies a^0 and a^1 of each $a \in \mathcal{A}$.

- Initially only the men in $\{a^0 : a \in \mathcal{A}\}$ are "active".

A linear time algorithm (K., 2012)

- Let $\tilde{G}_2$ be the graph on $\tilde{A}_2 \cup \mathcal{B}$ where $\tilde{A}_2$ consists of two copies a^0 and a^1 of each $a \in \mathcal{A}$.

- Initially only the men in $\{a^0 : a \in \mathcal{A}\}$ are “active”.

  - Active men propose and women dispose in $\tilde{G}_2$.

A linear time algorithm (K., 2012)

- Let $\tilde{G}_2$ be the graph on $\tilde{A}_2 \cup \mathcal{B}$ where $\tilde{A}_2$ consists of two copies a^0 and a^1 of each $a \in \mathcal{A}$.
- Initially only the men in $\{a^0 : a \in \mathcal{A}\}$ are “active”.
  - Active men propose and women dispose in $\tilde{G}_2$.
  - When any a_i^0 is rejected by all his neighbors:

A linear time algorithm (K., 2012)

- Let $\tilde{G}_2$ be the graph on $\tilde{A}_2 \cup \mathcal{B}$ where $\tilde{A}_2$ consists of two copies a^0 and a^1 of each $a \in \mathcal{A}$.

- Initially only the men in $\{a^0 : a \in \mathcal{A}\}$ are "active".
    - Active men propose and women dispose in $\tilde{G}_2$.
    - When any a_i^0 is rejected by all his neighbors:
        - introduce a_i^1 into the set of active vertices.

A linear time algorithm

- Termination condition: every a_i^j is either inactive or gets matched to some nbr.

A linear time algorithm

- Termination condition: every a_i^j is either inactive or gets matched to some nbr.

- Our algorithm is essentially Gale-Shapley algorithm on $\tilde{G}_2$.

A linear time algorithm

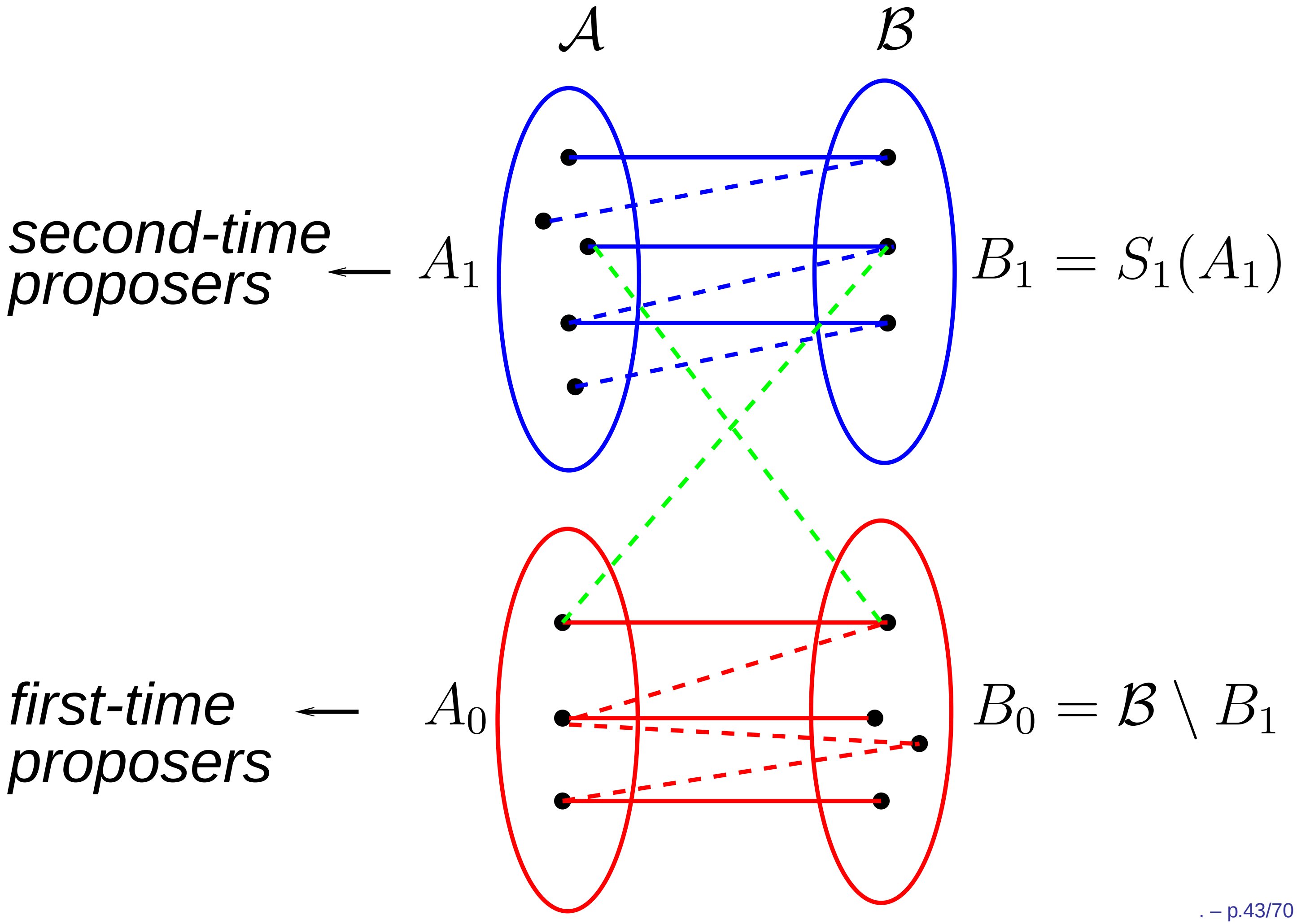
- Termination condition: every a_i^j is either inactive or gets matched to some nbr.

- Our algorithm is essentially Gale-Shapley algorithm on $\tilde{G}_2$.

  - Running time is $O(m+n)$, which is $O(m)$.

Properties of the output matching S_1

- $S_1 \subseteq (A_0 \times B_0) \cup (A_1 \times B_1)$.



Properties of the output matching S_1

- All unmatched vertices are in $A_1 \cup B_0$.

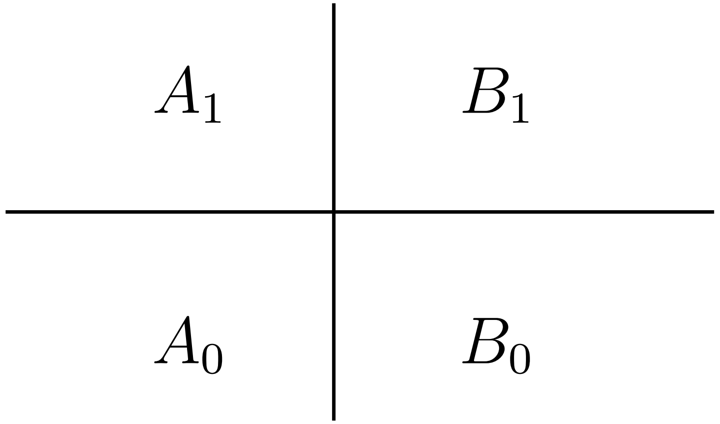
Properties of the output matching S_1

- All unmatched vertices are in $A_1 \cup B_0$.
- S_1 restricted to $A_i \cup B_i$ $(i = 0, 1)$ is stable.

A_1	B_1
A_0	B_0

Properties of the output matching S_1

- All unmatched vertices are in $A_1 \cup B_0$.
- S_1 restricted to $A_i \cup B_i$ $(i = 0, 1)$ is stable.



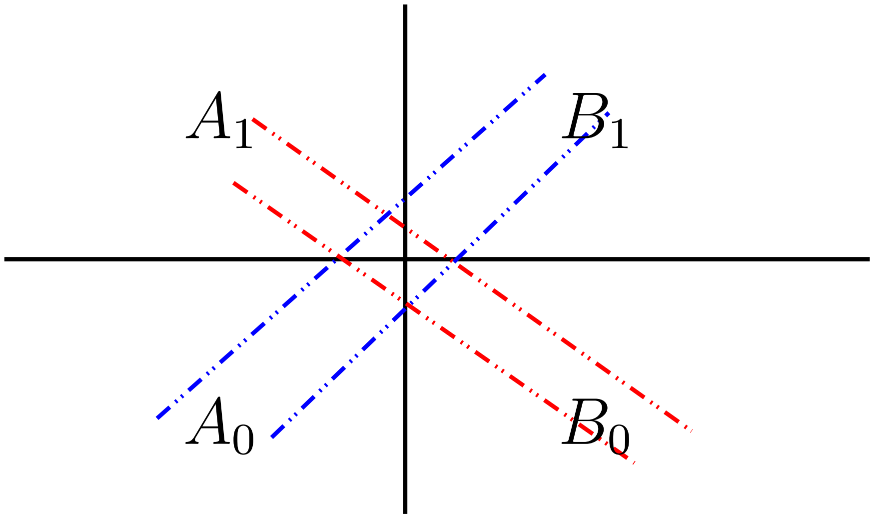
- Any blocking edge to S_1 has to be in $A_0 \times B_1$.

Partition of $\mathcal{A}$ and $\mathcal{B}$

- Every edge $(a, b) \in A_1 \times B_0$ is negative wrt S_1.

Partition of $\mathcal{A}$ and $\mathcal{B}$

- Every edge $(a, b) \in A_1 \times B_0$ is negative wrt S_1.



Popularity of S_1

- Consider the subgraph G_{S_1}.

Popularity of S_1

- Consider the subgraph G_{S_1}.
  - S_1 has the following properties in G_{S_1}:

Popularity of S_1

- Consider the subgraph G_{S_1}.
  - S_1 has the following properties in G_{S_1}:
    - no alternating cycle has a blocking edge.

Popularity of S_1

- Consider the subgraph G_{S_1}.
  - S_1 has the following properties in G_{S_1}:
    - no alternating cycle has a blocking edge.
    - no alternating path with a free endpoint has a blocking edge.

Popularity of S_1

- Consider the subgraph G_{S_1}.
  - S_1 has the following properties in G_{S_1}:
    - no alternating cycle has a blocking edge.
    - no alternating path with a free endpoint has a blocking edge.
    - no alternating path has 2 blocking edges.

Size of the matching S_1

- There is *no* augmenting path wrt S_1 in G_{S_1}.

Size of the matching S_1

- There is *no* augmenting path wrt S_1 in G_{S_1}.

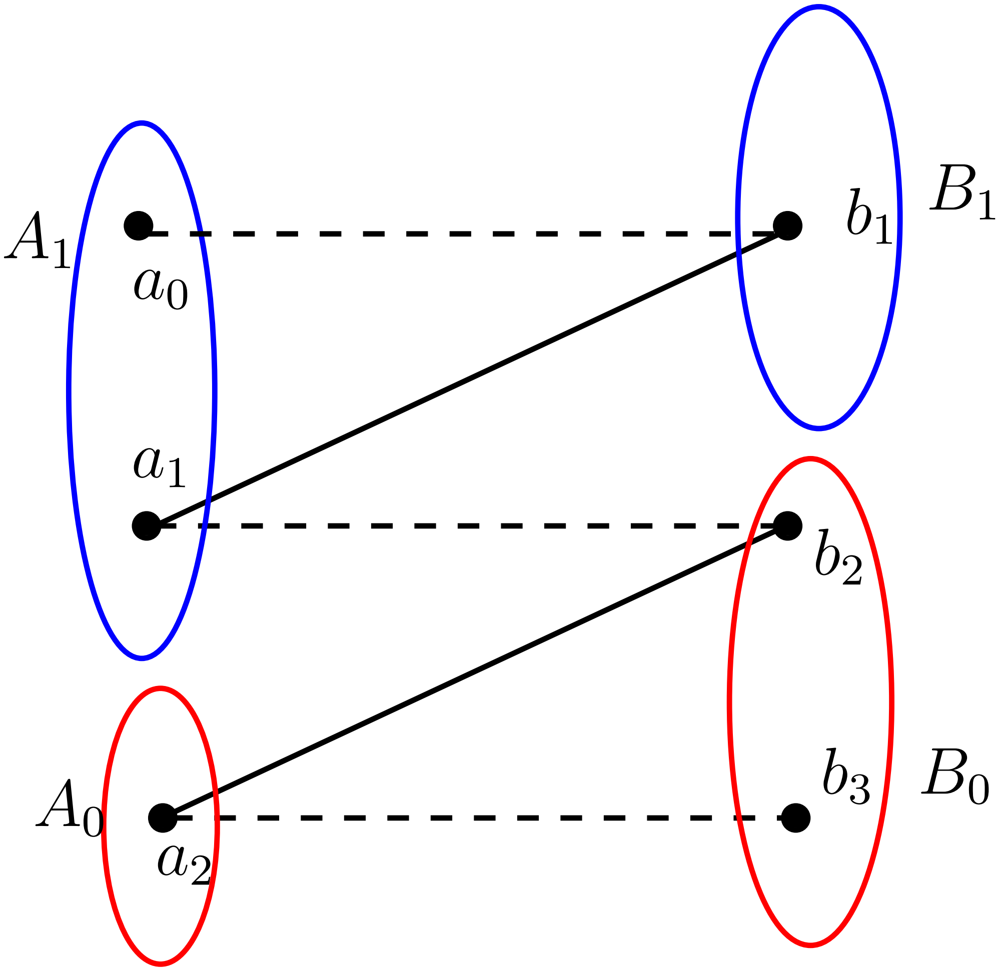
- Thus S_1 is a maximum size popular matching.

Size of the matching S_1

- There is *no* augmenting path wrt S_1 in G_{S_1}.
- Thus S_1 is a maximum size popular matching.
- What about $|S_1|$ in terms of $|M_{max}|$?

Size of the matching S_1

- Any augmenting path wrt S_1 in G has size ≥ 5:



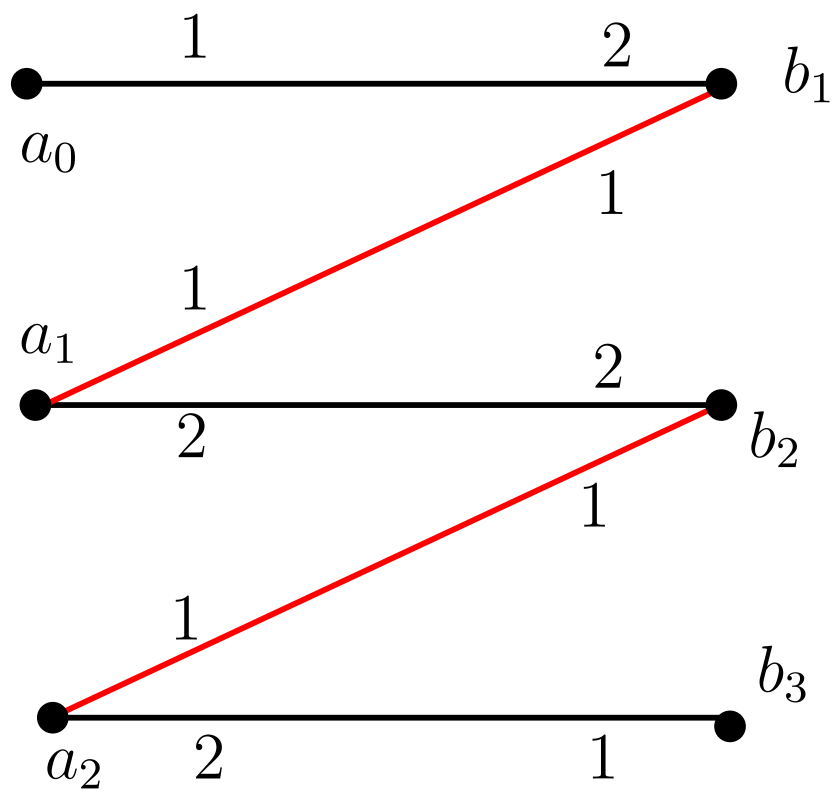
Size of the matching S_1

- Any augmenting path wrt S_1 in G has size $\geq 5 \Rightarrow$

Size of the matching S_1

- Any augmenting path wrt S_1 in G has size $\geq 5 \Rightarrow$
    - $|S_1| \geq \frac{2}{3}|M_{max}|$.

A tight example for the $2/3$ bound



- $|S_1| = 2$ while $|M_{max}| = 3$.

Larger size matchings

- Trade-off between popularity and size?

Larger size matchings

- Trade-off between popularity and size?
- Unpopularity factor $u(\cdot)$

Larger size matchings

- Trade-off between popularity and size?
- Unpopularity factor $u(\cdot)$
  - define $\delta(M, M')$ as the following ratio:

    $|\{$vertices that prefer $M'\}|/|\{$vertices that prefer $M\}|$

Larger size matchings

- Trade-off between popularity and size?
- Unpopularity factor $u(\cdot)$
  - define $\delta(M, M')$ as the following ratio:

    $|\{$vertices that prefer $M'\}|/|\{$vertices that prefer $M\}|$
  - $u(M) = \max_{M'} \delta(M, M')$.

Larger size matchings

- Trade-off between popularity and size?
- Unpopularity factor $u(\cdot)$
  - define $\delta(M, M')$ as the following ratio:

    |{vertices that prefer M'}|/|{vertices that prefer M}|
  - $u(M) = \max_{M'} \delta(M, M')$.

$u(M) = \beta \Rightarrow$ for every matching M' we have:
$|\{$vertices that prefer $M'\}| \leq \beta \cdot |\{$vertices that prefer $M\}|$.

Popularity vs Size

- M is popular $\Leftrightarrow u(M) \leq 1$.

Popularity vs Size

- M is popular $\Leftrightarrow u(M) \leq 1$.
  - We can find a matching S_1 with $u(S_1) \leq 1$ and $|S_1| \geq \frac{2}{3}|M_{max}|$.

Popularity vs Size

- M is popular $\Leftrightarrow u(M) \leq 1$.
  - We can find a matching S_1 with $u(S_1) \leq 1$ and $|S_1| \geq \frac{2}{3}|M_{max}|$.
  - For every integer $k \geq 2$, can we find a matching S_k with $u(S_k) \leq k-1$ and $|S_k| \geq \frac{k}{k+1}|M_{max}|$?

Popularity vs Size

- M is popular $\Leftrightarrow u(M) \leq 1$.
  - We can find a matching S_1 with $u(S_1) \leq 1$ and $|S_1| \geq \frac{2}{3}|M_{max}|$.
  - For every integer $k \geq 2$, can we find a matching S_k with $u(S_k) \leq k-1$ and $|S_k| \geq \frac{k}{k+1}|M_{max}|$?
- Is there an $M^* \equiv$ a maximum cardinality matching s.t. for each max cardinality matching M: $M^* \succeq M$?

Extending the 2-level algorithm

- For any integer $k \geq 2$, we can extend the 2-level algorithm to k levels.

Extending the 2-level algorithm

- For any integer $k \geq 2$, we can extend the 2-level algorithm to k levels.
    - So the graph becomes $\tilde{G}_k$ on $\tilde{\mathcal{A}}_k \cup \mathcal{B}$.

Extending the 2-level algorithm

- For any integer $k \geq 2$, we can extend the 2-level algorithm to k levels.
  - So the graph becomes $\tilde{G}_k$ on $\tilde{\mathcal{A}}_k \cup \mathcal{B}$.
    - $\tilde{\mathcal{A}}_k$ has k copies $a^0, a^1, \ldots, a^{k-1}$ of each $a \in \mathcal{A}$. (a^i is a level i vertex)

Extending the 2-level algorithm

- For any integer $k \geq 2$, we can extend the 2-level algorithm to k levels.
  - So the graph becomes $\tilde{G}_k$ on $\tilde{\mathcal{A}}_k \cup \mathcal{B}$.
    - $\tilde{\mathcal{A}}_k$ has k copies $a^0, a^1, \ldots, a^{k-1}$ of each $a \in \mathcal{A}$. (a^i is a level i vertex)
    - For each $a \in \mathcal{A}$: at most one of $a^0, a^1, \ldots, a^{k-1}$ is active at any point.

The k-level algorithm

- Corresponding to each edge (a, b) in G:

The k-level algorithm

- Corresponding to each edge (a, b) in G:
  - we have k edges (a^i, b) for $i = 0, \ldots, k-1$ in $\tilde{G}_k$.

The k-level algorithm

- Corresponding to each edge (a, b) in G:
  - we have k edges (a^i, b) for $i = 0, \ldots, k-1$ in $\tilde{G}_k$.
- In $\tilde{G}_k$, the preference list of any $b \in \mathcal{B}$:

The k-level algorithm

- Corresponding to each edge (a, b) in G:
  - we have k edges (a^i, b) for $i = 0, \ldots, k-1$ in $\tilde{G}_k$.
- In $\tilde{G}_k$, the preference list of any $b \in \mathcal{B}$:
  - level $(k-1)$ neighbors

The k-level algorithm

- Corresponding to each edge (a, b) in G:
  - we have k edges (a^i, b) for $i = 0, \ldots, k-1$ in $\tilde{G}_k$.

- In $\tilde{G}_k$, the preference list of any $b \in \mathcal{B}$:
  - level $(k-1)$ neighbors
  - then level $(k-2)$ neighbors, ... and so on ...,

The k-level algorithm

- Corresponding to each edge (a, b) in G:
  - we have k edges (a^i, b) for $i = 0, \ldots, k-1$ in $\tilde{G}_k$.
- In $\tilde{G}_k$, the preference list of any $b \in \mathcal{B}$:
  - level $(k-1)$ neighbors
  - then level $(k-2)$ neighbors, ... and so on ...,
  - and at the bottom are level 0 neighbors.

The k-level algorithm

- Set level 0 men to be active and set higher level men to be inactive.

The k-level algorithm

- Set level 0 men to be active and set higher level men to be inactive.
- Essentially Gale-Shapley with the active men proposing and women disposing:

The k-level algorithm

- Set level 0 men to be active and set higher level men to be inactive.

- Essentially Gale-Shapley with the active men proposing and women disposing:
  - $i < k - 1$: if a^i is rejected by all his neighbors, then a^{i+1} becomes active.

The k-level algorithm

- Set level 0 men to be active and set higher level men to be inactive.

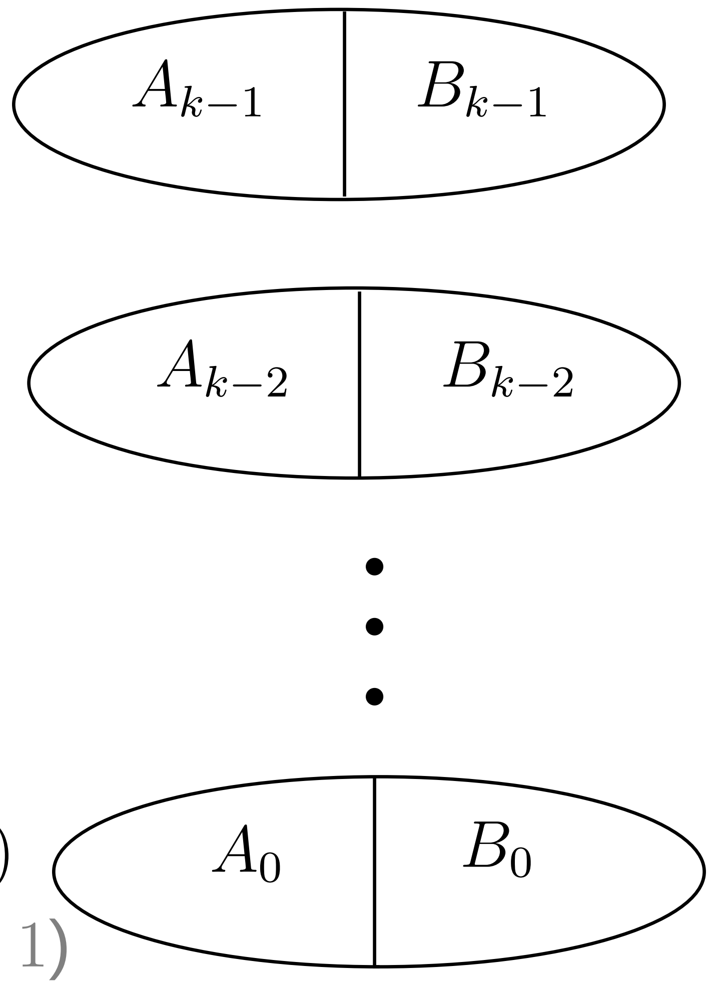
- Essentially Gale-Shapley with the active men proposing and women disposing:

  - $i < k - 1$: if a^i is rejected by all his neighbors, then a^{i+1} becomes active.

- Let S_{k-1} be the matching returned by this algorithm.

The partition of $\mathcal{A}$ and $\mathcal{B}$

- $A_i = \{a \in \mathcal{A}$ such that a is in level i at the end$\}$.

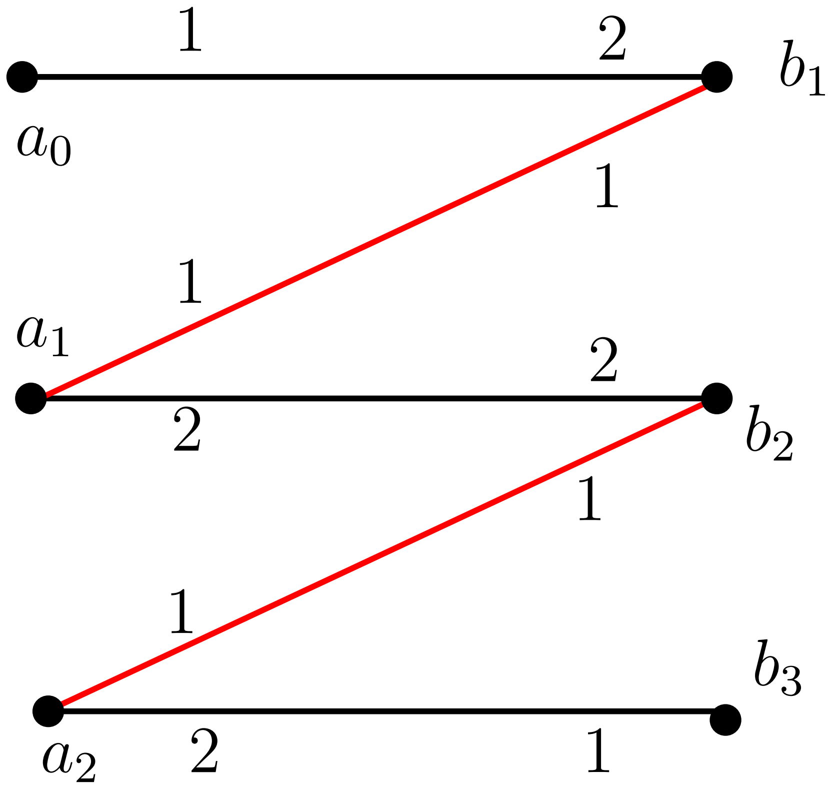


- $B_i = S_{k-1}(A_i)$

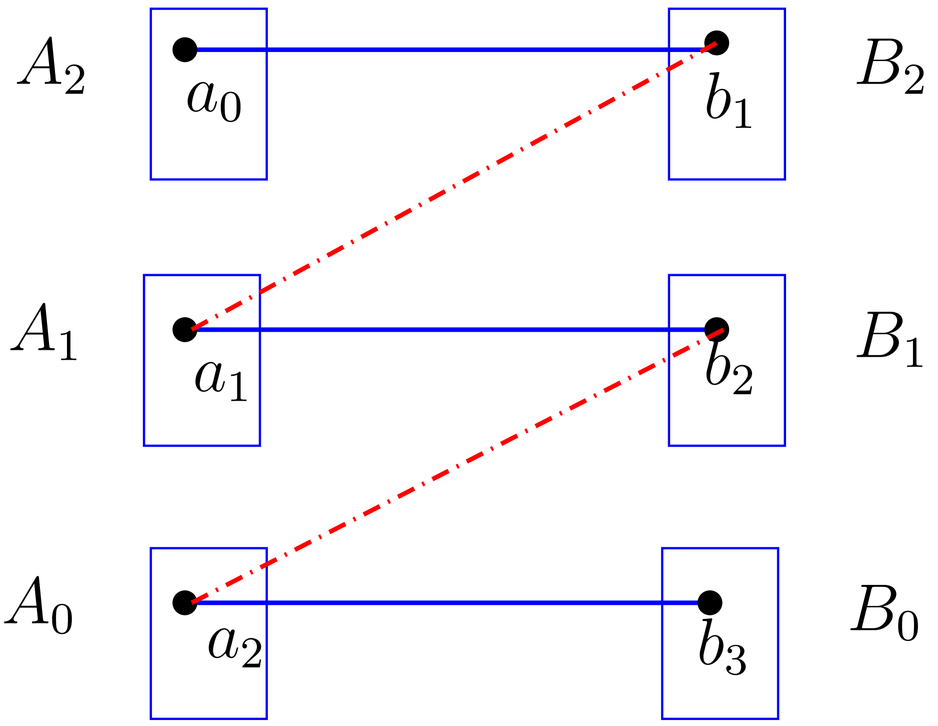
(for $1 \leq i \leq k-1$)

The 3-level algorithm

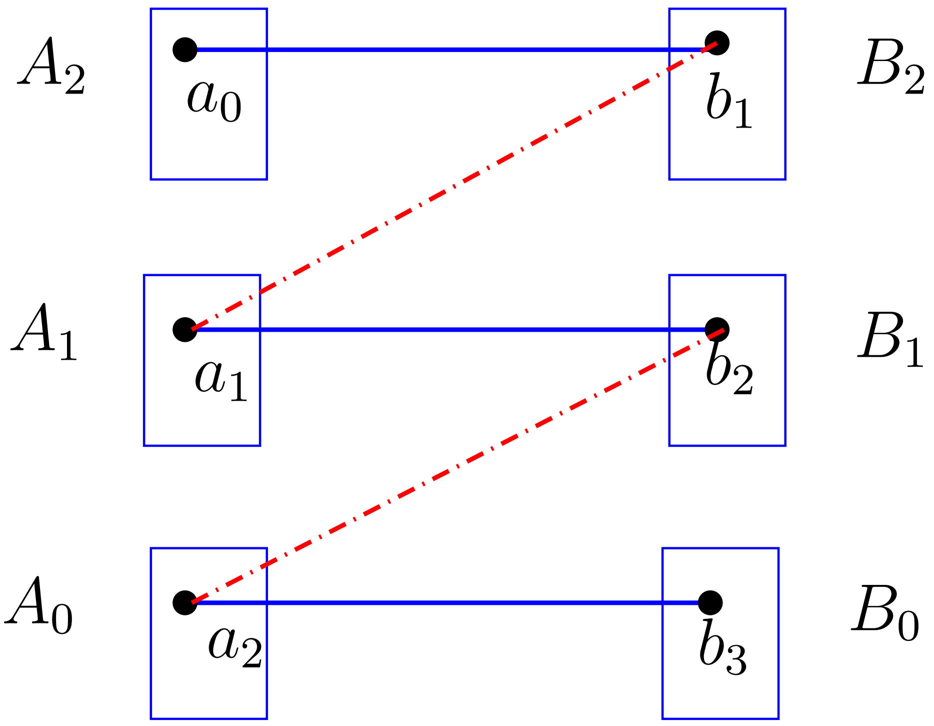
- Say we run the 3-level algorithm on our tight example for the 2-level algorithm ...



In the 3-level algorithm



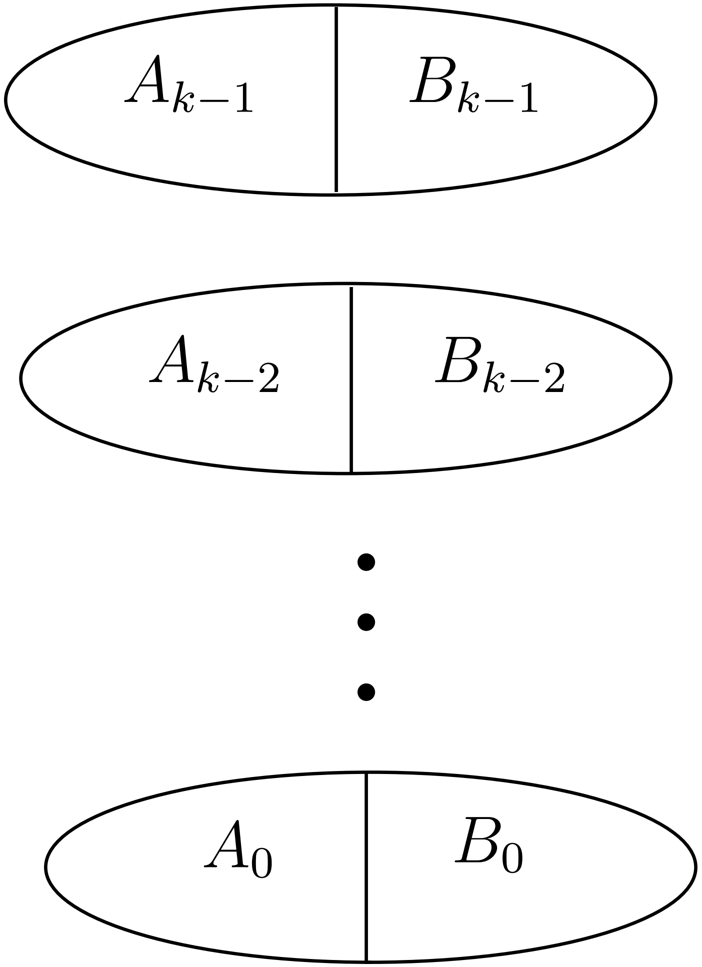
In the 3-level algorithm



- The matching $S_2 = \{(a_0, b_1), (a_1, b_2), (a_2, b_3)\}$ is output by the 3-level algorithm.

Properties of the matching S_{k-1}

- $S_{k-1} \subseteq (A_{k-1} \times B_{k-1}) \cup (A_{k-2} \times B_{k-2}) \cup \cdots \cup (A_0 \times B_0)$.



Properties of the matching S_{k-1}

- Every unmatched vertex is in $A_{k-1} \cup B_0$.

Properties of the matching S_{k-1}

- Every unmatched vertex is in $A_{k-1} \cup B_0$.

- For all i: no edge in G between $A_{k-1} \cup \cdots \cup A_{i+1}$ and $B_{i-1} \cup \cdots \cup B_0$.

Properties of the matching S_{k-1}

- Every unmatched vertex is in $A_{k-1} \cup B_0$.
- For all i: no edge in G between $A_{k-1} \cup \cdots \cup A_{i+1}$ and $B_{i-1} \cup \cdots \cup B_0$.
  - any augmenting path wrt S_{k-1} has length $\geq 2k+1$.

Properties of the matching S_{k-1}

- Every unmatched vertex is in $A_{k-1} \cup B_0$.
- For all i: no edge in G between $A_{k-1} \cup \cdots \cup A_{i+1}$ and $B_{i-1} \cup \cdots \cup B_0$.
  - any augmenting path wrt S_{k-1} has length $\geq 2k+1$.
    - hence $|S_{k-1}| \geq \frac{k}{k+1}|M_{max}|$.

Unpopularity of S_{k-1}

- Consider the subgraph $G_{S_{k-1}}$.

Unpopularity of S_{k-1}

- Consider the subgraph $G_{S_{k-1}}$.
  - S_{k-1} has the following properties in this graph:

Unpopularity of S_{k-1}

- Consider the subgraph $G_{S_{k-1}}$.
  - S_{k-1} has the following properties in this graph:
    - no alternating cycle has a blocking edge.

Unpopularity of S_{k-1}

- Consider the subgraph $G_{S_{k-1}}$.
  - S_{k-1} has the following properties in this graph:
    - no alternating cycle has a blocking edge.
    - no alternating path with a free endpoint has a blocking edge.

Unpopularity of S_{k-1}

- Consider the subgraph $G_{S_{k-1}}$.
  - S_{k-1} has the following properties in this graph:
    - no alternating cycle has a blocking edge.
    - no alternating path with a free endpoint has a blocking edge.
    - no alternating path has k blocking edges.

Trade-off between size and unpopularity

- This implies that $u(S_{k-1}) \leq k - 1$.

Trade-off between size and unpopularity

- This implies that $u(S_{k-1}) \leq k-1$.

- Thus for any $k \geq 2$, there exists a matching S_{k-1} s.t. $u(S_{k-1}) \leq k-1$ and $|S_{k-1}| \geq \frac{k}{k+1}|M_{max}|$.

Trade-off between size and unpopularity

- This implies that $u(S_{k-1}) \leq k-1$.

- Thus for any $k \geq 2$, there exists a matching S_{k-1} s.t. $u(S_{k-1}) \leq k-1$ and $|S_{k-1}| \geq \frac{k}{k+1}|M_{max}|$.

- S_{k-1} can be computed in $O(mk)$ time.

The boundary cases

- $k = 2$: S_1 is a maximum size popular matching

The boundary cases

- $k = 2$: S_1 is a maximum size popular matching

- $k = n_0$:

The boundary cases

- $k = 2$: S_1 is a maximum size popular matching
- $k = n_0$:
  - $|S_{n_0-1}| \geq \frac{n_0}{n_0+1}|M_{max}|$ and $|M_{max}| \leq n_0$, so $|S_{n_0-1}| = |M_{max}|$.

The boundary cases

- $k = 2$: S_1 is a maximum size popular matching
- $k = n_0$:
  - $|S_{n_0-1}| \geq \frac{n_0}{n_0+1}|M_{max}|$ and $|M_{max}| \leq n_0$, so $|S_{n_0-1}| = |M_{max}|$.
  - for any max cardinality matching M: $S_{n_0-1} \succeq M$.

In general graphs

- Input $G = (V, E)$: a general graph with strict 2-sided preference lists

In general graphs

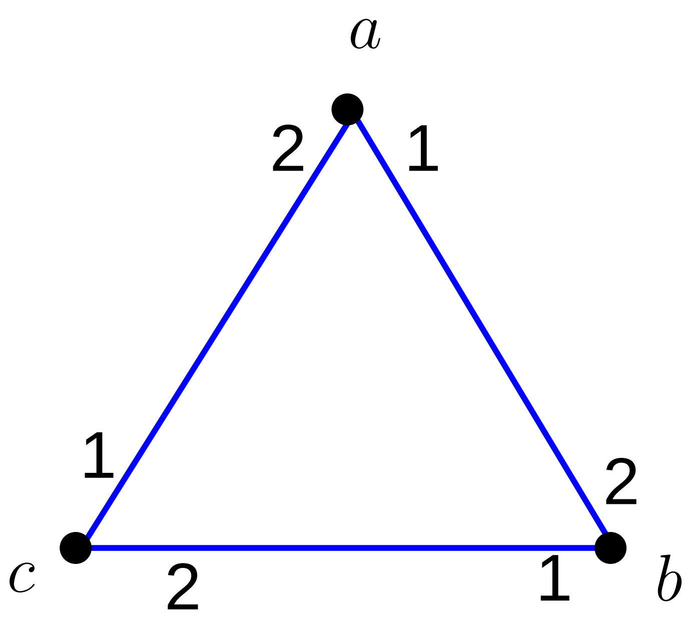
- Input $G = (V, E)$: a general graph with strict 2-sided preference lists

- Stable matchings need not always exist in non-bipartite graphs.

In general graphs

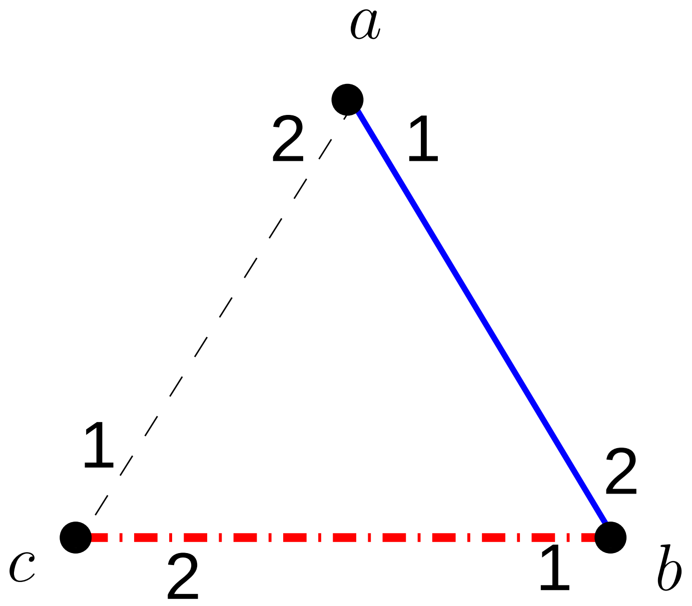
- Input $G = (V, E)$: a general graph with strict 2-sided preference lists

- Stable matchings need not always exist in non-bipartite graphs.



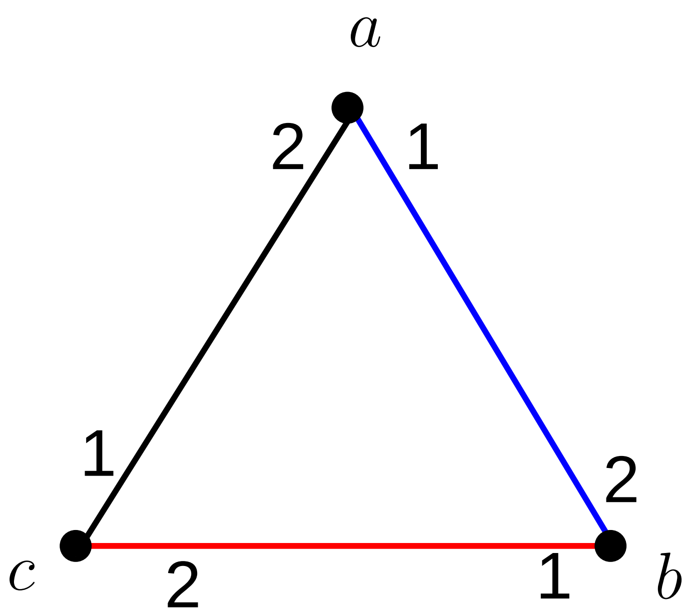
In general graphs

- Input $G = (V, E)$: a general graph with strict 2-sided preference lists
- Stable matchings need not always exist in non-bipartite graphs: every matching here has a "blocking edge".



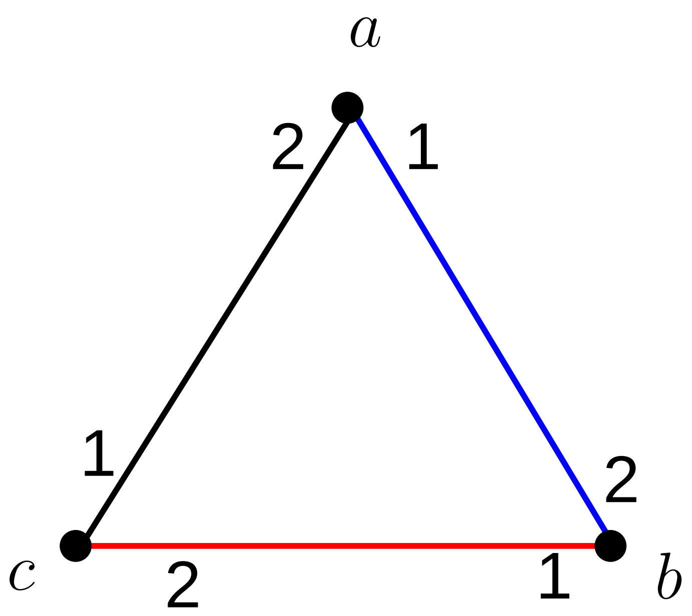
In general graphs

- In fact, this instance has no popular matching either.



In general graphs

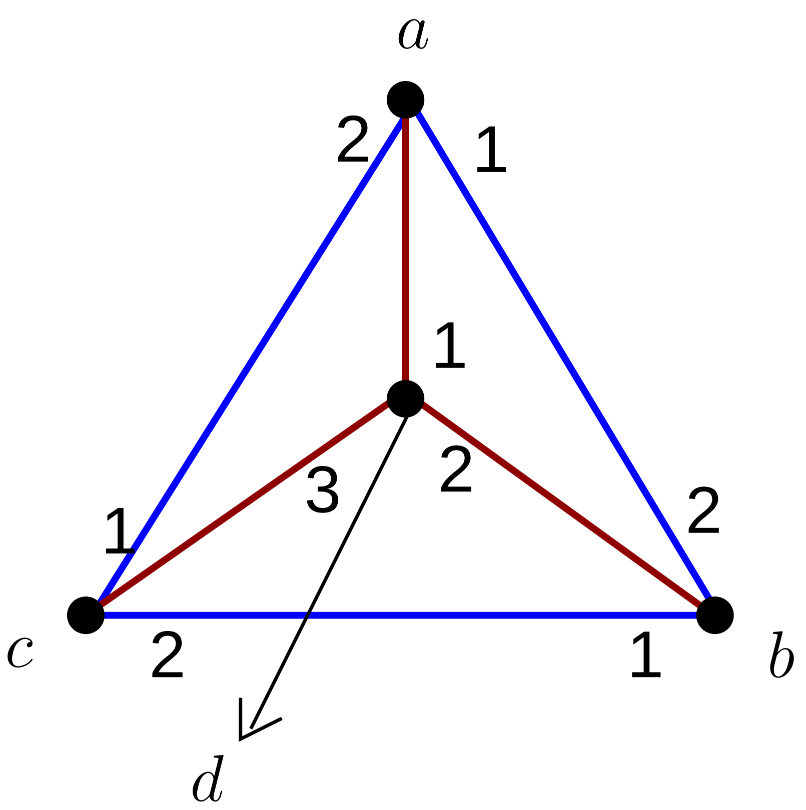
- In fact, this instance has no popular matching either.



- We have $M_1 \prec M_2 \prec M_3 \prec M_1$ here, where $M_1 = \{(a,b)\}$, $M_2 = \{(b,c)\}$, and $M_3 = \{(a,c)\}$.

In general graphs

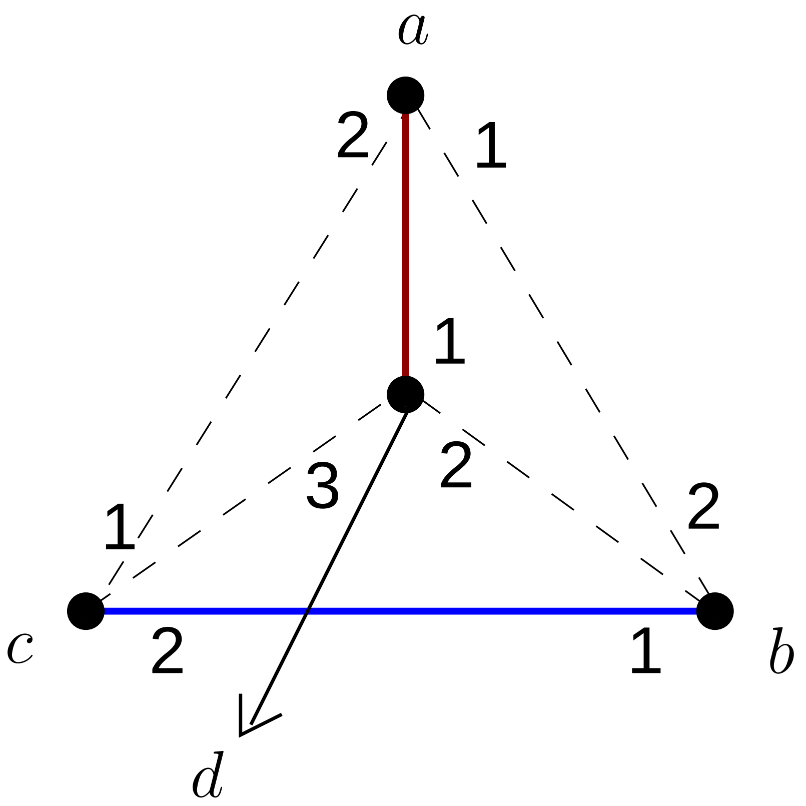
- An instance with no stable matching but with popular matchings:



- d is the least preferred neighbor for a, b, c.

In general graphs

- An instance with no stable matching but with popular matchings:



- $\{(a, d), (b, c)\}$ is popular.

In general graphs (Huang and K., 2011)

- There is always a matching M such that $u(M)$ is $O(\log n)$.

In general graphs (Huang and K., 2011)

- There is always a matching M such that $u(M)$ is $O(\log n)$.

    - Such a matching can be computed in linear time.

In general graphs (Huang and K., 2011)

- There is always a matching M such that $u(M)$ is $O(\log n)$.
    - Such a matching can be computed in linear time.
- Computing a *least* unpopularity factor matching is NP-hard.

In general graphs (Huang and K., 2011)

- There is always a matching M such that $u(M)$ is $O(\log n)$.
  - Such a matching can be computed in linear time.
- Computing a *least* unpopularity factor matching is NP-hard.
- *Open problem*: complexity of determining if G admits a popular matching or not.

Thank you!