Popular Matchings

Kavitha Telikepalli

(Tata Institute of Fundamental Research, Mumbai)

COST Action IC1205 on Computational Social Choice

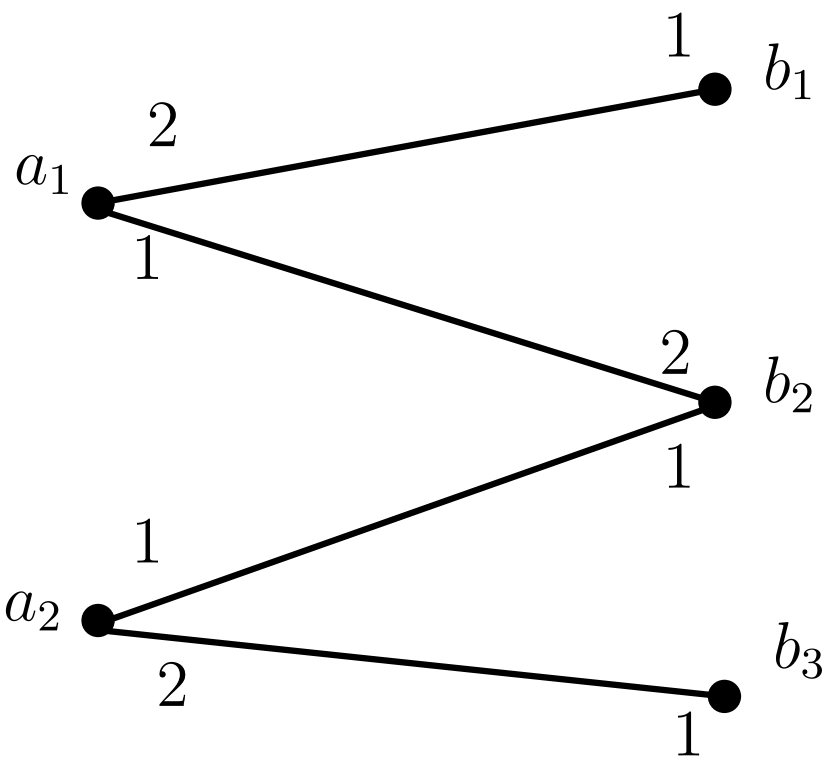
and MATCH-UP 2015, University of Glasgow.

Our problem

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

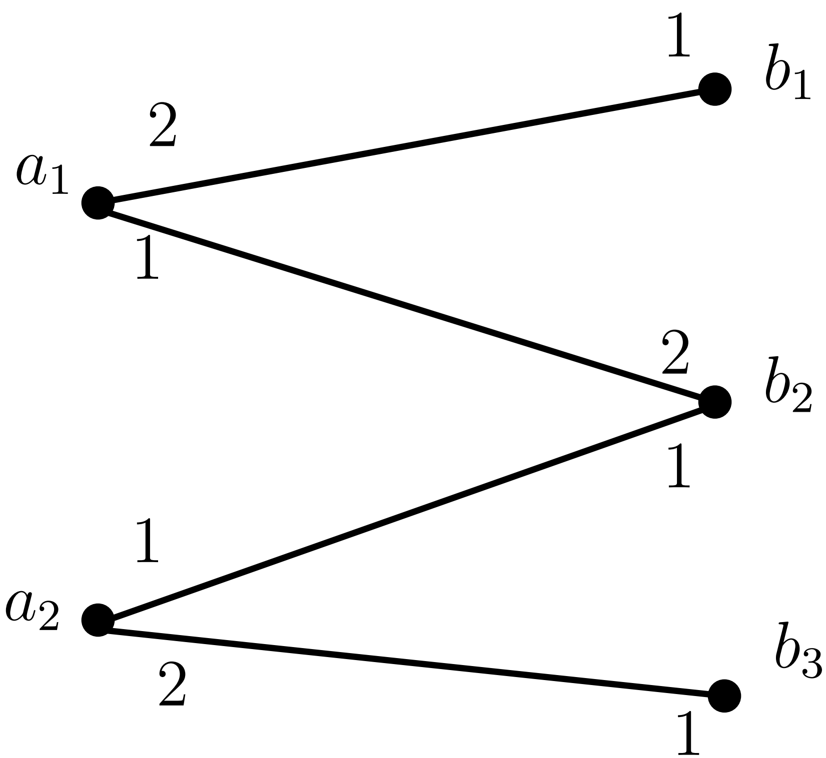
Our problem

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



Our problem

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



- $\mathcal{A}$: a set of students; $\mathcal{B}$: a 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 an “optimal” matching M in G.

The input

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

- Problem: compute an "optimal" matching M in G.

  - there are no vertices $a \in \mathcal{A}$ and $b \in \mathcal{B}$ who are *better-off* by being matched to each other

The input

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

- Problem: compute an "optimal" matching M in G.

  - there are no vertices $a \in \mathcal{A}$ and $b \in \mathcal{B}$ who are *better-off* by being matched to each other

    - M is a stable matching.

Price of stability

- From a global point of view, M_{max} is the optimal matching.

Price of stability

- From a global point of view, M_{max} is the optimal matching.
- Size of a stable matching:

Price of stability

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

Price of stability

- From a global point of view, 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*: (Gärdenfors 1975)

Popular matchings

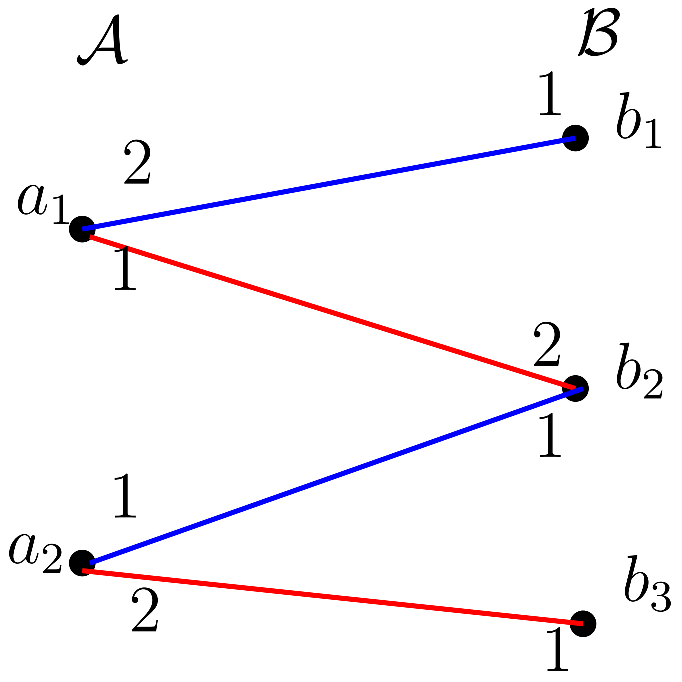
- A new notion of optimality that is a compromise between M_{max} and a stable matching?

- A notion based on *popularity*: (Gärdenfors 1975)

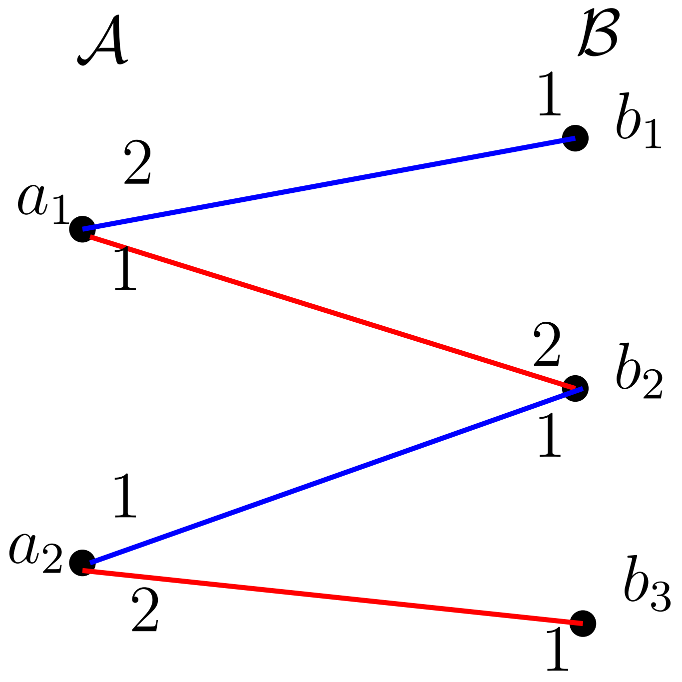
  - matching M_1 is *more popular* than matching M_2 if

$\#$ of vertices that prefer $M_1 > \#$ of vertices that prefer M_2.

An example

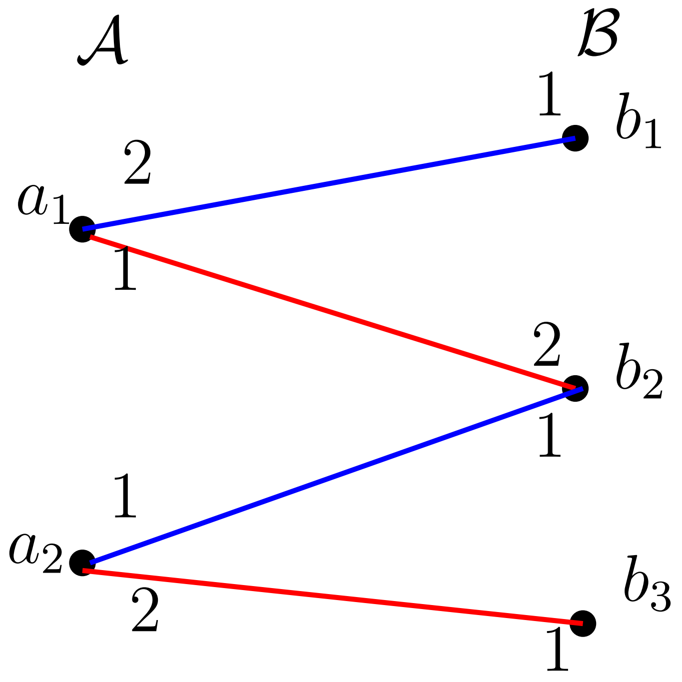


An example



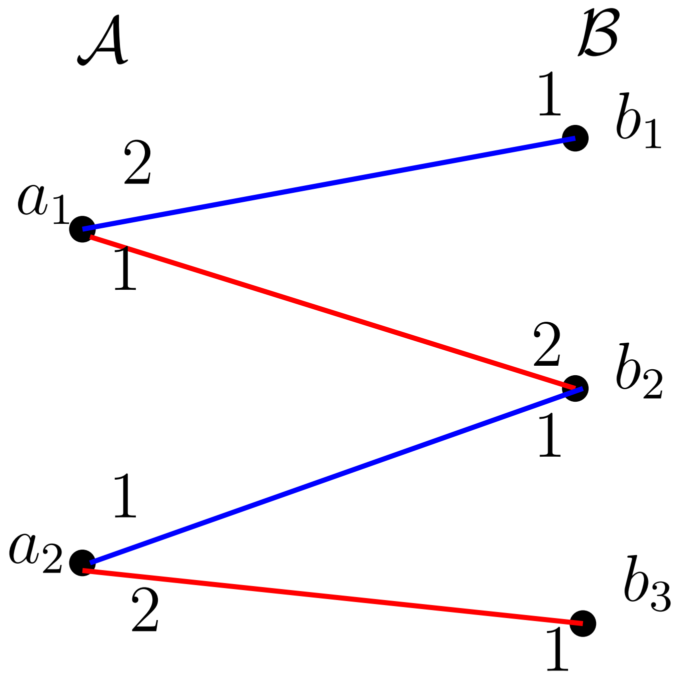
- a_1 and b_3 prefer the red matching

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.

- M is popular if there is *no* M' such that $M' \succ M$.

Popular matchings

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

- M is popular if there is *no* M' such that $M' \succ M$.

M is popular $\Rightarrow$ for every matching M' we have:
$\#$ of vertices that prefer $M' \leq \#$ of vertices that prefer M.

Popular matchings

- The "*more popular than*" relation is not transitive: we can have $M_1 \succ M_2 \succ M_3 \succ M_1$.

Popular matchings

- The "*more popular than*" relation is not transitive: we can have $M_1 \succ M_2 \succ M_3 \succ M_1$.

- Do popular matchings always exist in G?

Popular matchings

- The "*more popular than*" relation is not transitive: we can have $M_1 \succ M_2 \succ M_3 \succ M_1$.

- Do popular matchings always exist in G?

    - yes; in fact, every stable matching is popular.

Popular matchings

- The "*more popular than*" relation is not transitive: we can have $M_1 \succ M_2 \succ M_3 \succ M_1$.
- Do popular matchings always exist in G?
  - yes; in fact, every stable matching is popular.
    - thus $\{$stable matchings$\} \subseteq \{$popular matchings$\}$.

stable $\implies$ popular

- Comparing a stable matching S with any matching M:

stable $\Longrightarrow$ popular

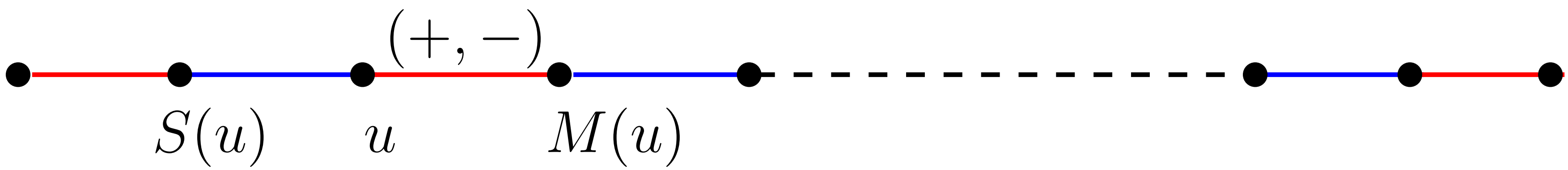
- Comparing a stable matching S with any matching M:

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

stable $\Longrightarrow$ popular

- Comparing a stable matching S with any matching M:

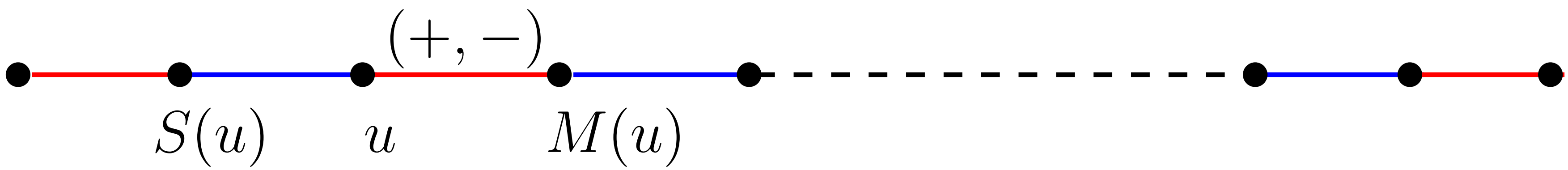
u prefers M to $S \Rightarrow M(u)$ has to prefer S to M.



stable $\Longrightarrow$ popular

- Comparing a stable matching S with any matching M:

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

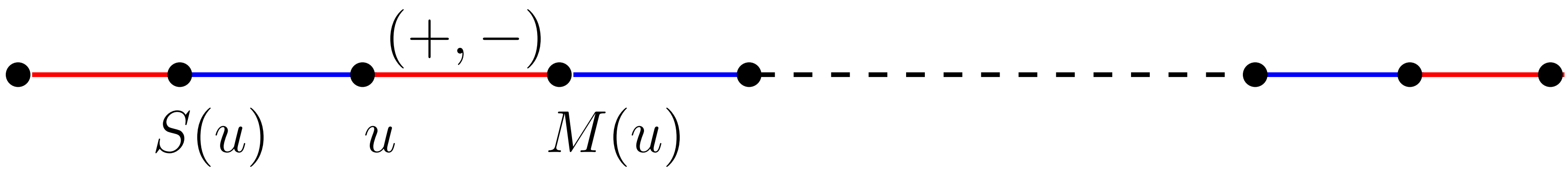


- Label red edges by $(+,+)$ / $(-,-)$ / $(+,-)$.

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.

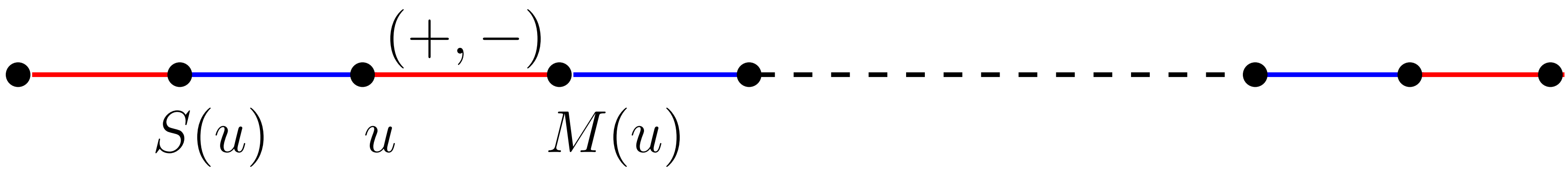


- Label red edges by $(+,+)$ / $(-,-)$ / $(+,-)$.
  - there is no edge that is labeled $(+,+)$:

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.



- Label red edges by $(+,+)$ / $(-,-)$ / $(+,-)$.
  - there is no edge that is labeled $(+,+)$:
    - so # of votes for $M \leq$ # of votes for S.

Stable matchings

- A stable matching is a *minimum* size popular matching.

Stable matchings

- A stable matching is a *minimum* size popular matching.
- Let S be a stable matching and let M be a smaller matching.

Stable matchings

- A stable matching is a *minimum* size popular matching.
- Let S be a stable matching and let M be a smaller matching.
- $|M| < |S|$, so $M \oplus S$ has an augmenting path p wrt M.

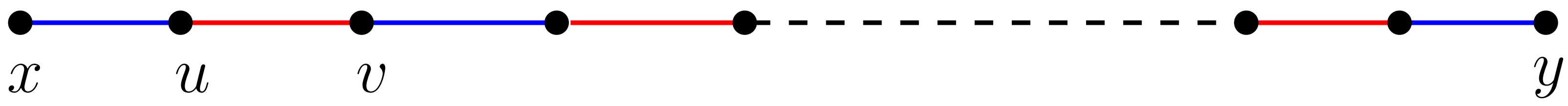
Stable matchings

- A stable matching is a *minimum* size popular matching.
- Let S be a stable matching and let M be a smaller matching.
- $|M| < |S|$, so $M \oplus S$ has an augmenting path p wrt M.
- *Claim:* $M \oplus p \succ M$.

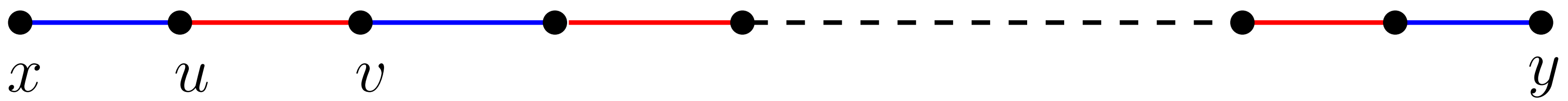
Stable matchings

- A stable matching is a *minimum* size popular matching.
- Let S be a stable matching and let M be a smaller matching.
- $|M| < |S|$, so $M \oplus S$ has an augmenting path p wrt M.
- *Claim:* $M \oplus p \succ M$.
  - thus M is unpopular

The alternating path p

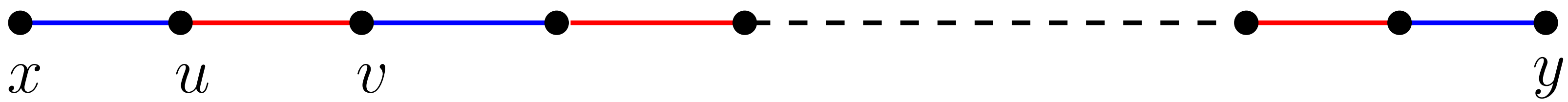


The alternating path p



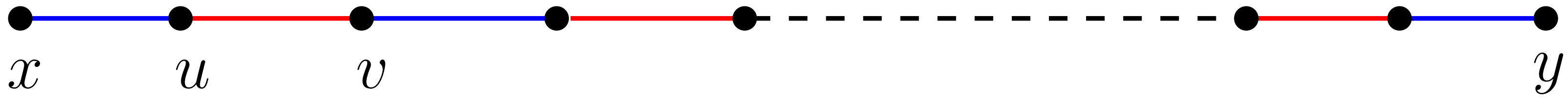
- red: edges of M; blue: edges of S.

The alternating path p



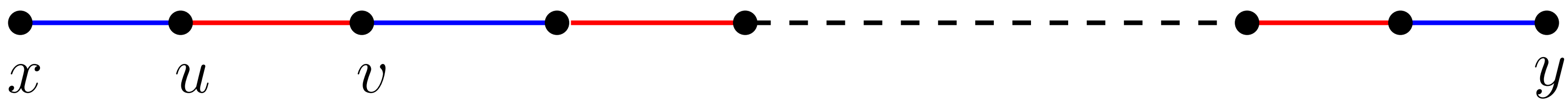
- red: edges of M; blue: edges of S.
- both x and y prefer S to M.

The alternating path p



- red: edges of M; blue: edges of S.
- both x and y prefer S to M.
- for every M-edge (u, v) in p:
  u prefers M to $S \Rightarrow v$ prefers S to M.

The alternating path p

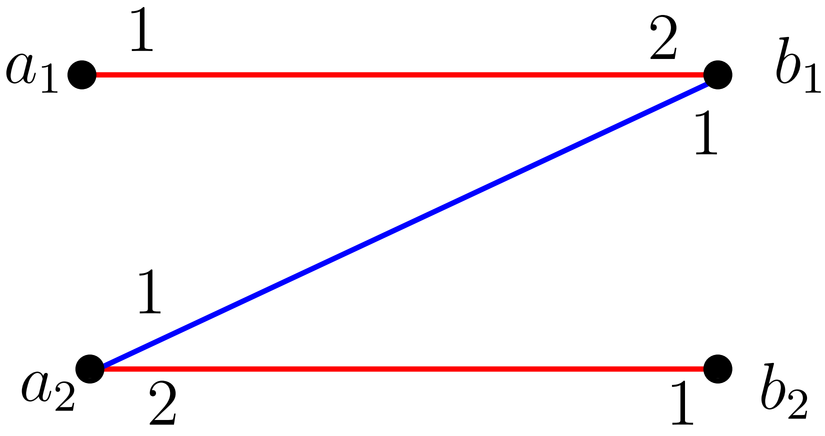


- red: edges of M; blue: edges of S.
- both x and y prefer S to M.
- for every M-edge (u, v) in p:
  u prefers M to $S \Rightarrow v$ prefers S to M.

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

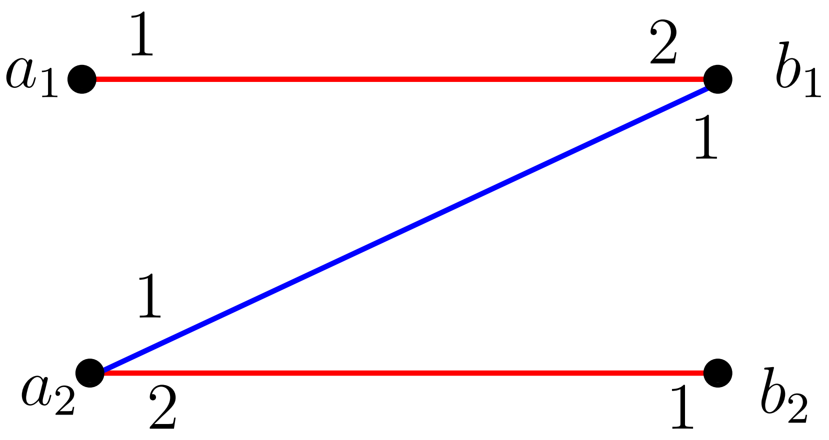
Min vs max size popular matchings

- 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| $> |M_{max}|/2$ always?

Some questions

- Is |max size popular matching| $> |M_{max}|/2$ always?
- Structural characterization of popular matchings?

Some questions

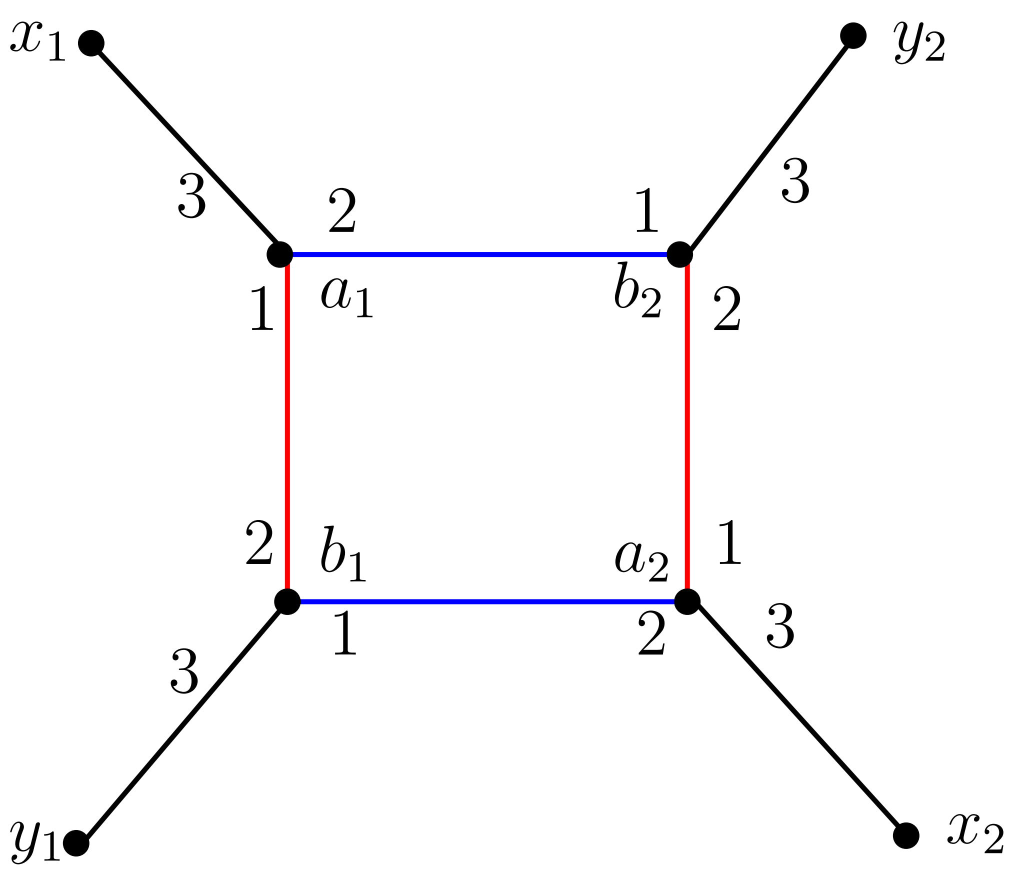
- Is |max size popular matching| $> |M_{max}|/2$ always?
- Structural characterization of popular matchings?
- Structural characterization of maximum size popular matchings?

Some questions

- Is |max size popular matching| $> |M_{max}|/2$ always?
- Structural characterization of popular matchings?
- Structural characterization of maximum size popular matchings?
- Can a maximum size popular matching be efficiently computed?

An interesting example

- Popular matchings of size 2 and size 4; none of size 3.



Some questions

- Is |max size popular matching| $> |M_{max}|/2$ always?
- Structural characterization of popular matchings? $\surd$
- Structural characterization of maximum size popular matchings?
- Can a maximum size popular matching be efficiently computed?

The subgraph G_M

- Let M be any matching in G.

The subgraph G_M

- Let M be any matching in G.
    - label edges outside M by $(+,+)$ / $(-,-)$ / $(+,-)$

The subgraph G_M

- Let M be any matching in G.
  - label edges outside M by $(+,+)$ / $(-,-)$ / $(+,-)$
  - call an edge (u,v) *negative* wrt M if it is labeled $(-,-)$

The subgraph G_M

- Let M be any matching in G.
  - label edges outside M by $(+,+)$ / $(-,-)$ / $(+,-)$
  - call an edge (u,v) *negative* wrt M if it is labeled $(-,-)$
    - u prefers $M(u)$ to v *and* v prefers $M(v)$ to u.

The subgraph G_M

- Let M be any matching in G.
  - label edges outside M by $(+,+)$ / $(-,-)$ / $(+,-)$
  - call an edge (u,v) *negative* wrt M if it is labeled $(-,-)$
    - u prefers $M(u)$ to v *and* v prefers $M(v)$ to u.
- Delete from G all negative edges wrt M — call this graph G_M.

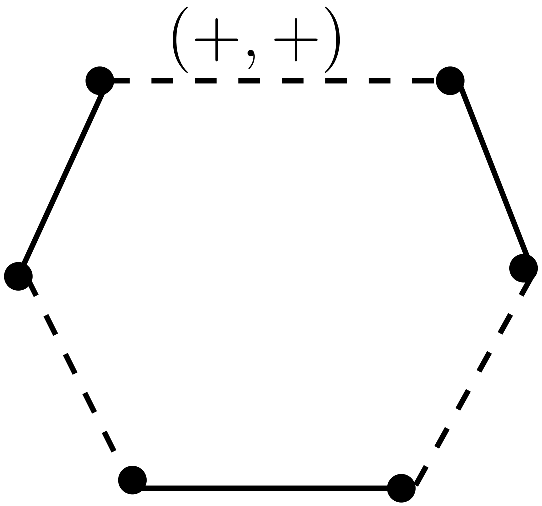
Necessary conditions for popularity

- If M is popular then M has to satisfy these 3 properties in G_M:

Necessary conditions for popularity

- If M is popular then M has to satisfy these 3 properties in G_M:

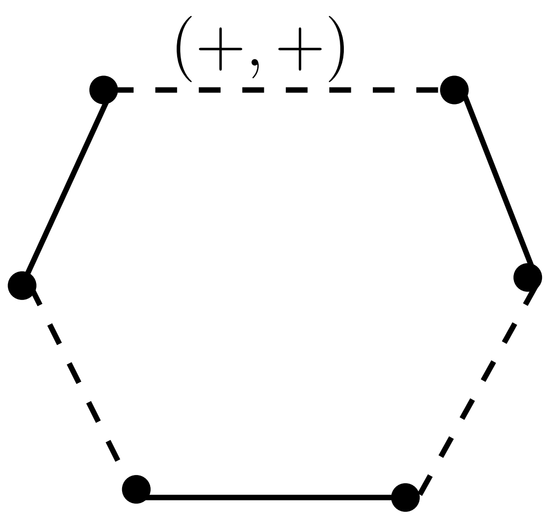
(1) no alternating cycle C with a $(+,+)$ edge, i.e., a *blocking edge*.



Necessary conditions for popularity

- If M is popular then M has to satisfy these 3 properties in G_M:

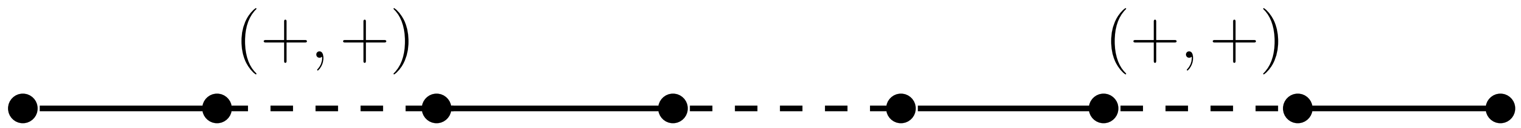
(1) no alternating cycle C with a $(+,+)$ edge, i.e., a *blocking edge*.



- otherwise $M \oplus C \succ M$.

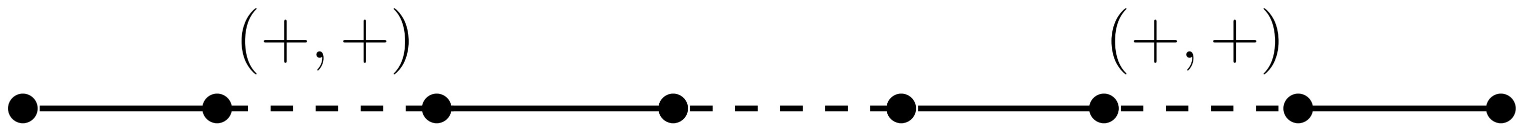
Necessary conditions for popularity

(2) no alternating path ρ with 2 blocking edges.



Necessary conditions for popularity

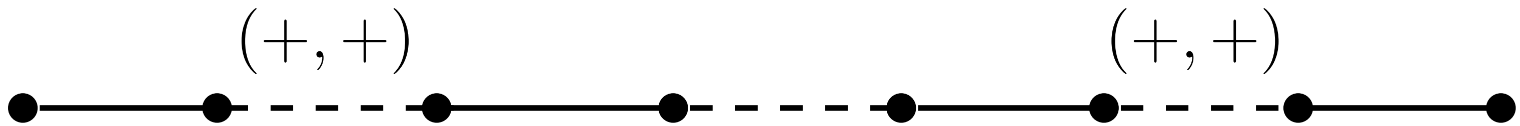
(2) no alternating path ρ with 2 blocking edges.



- otherwise $M \oplus \rho \succ M$.

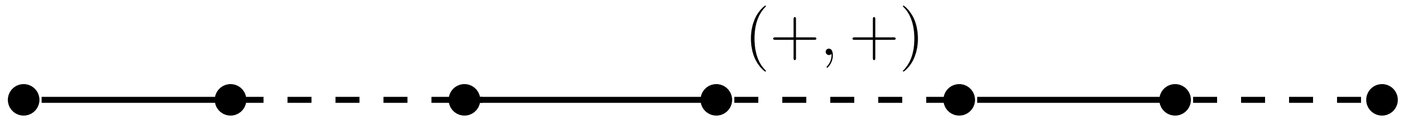
Necessary conditions for popularity

(2) no alternating path ρ with 2 blocking edges.



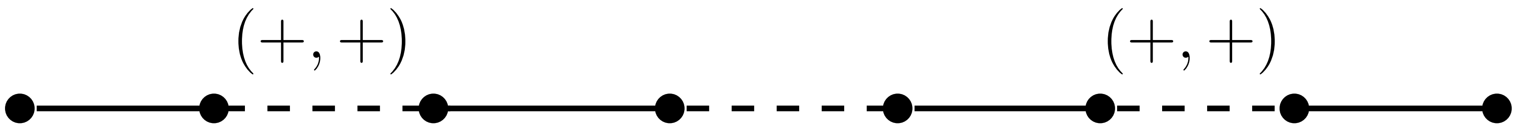
- otherwise $M \oplus \rho \succ M$.

(3) no alternating path p with an unmatched endpoint and a blocking edge.



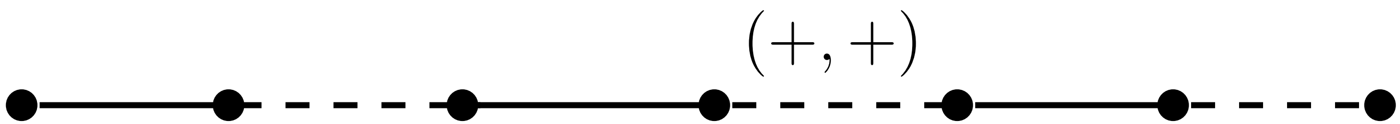
Necessary conditions for popularity

(2) no alternating path ρ with 2 blocking edges.



- otherwise $M \oplus \rho \succ M$.

(3) no alternating path p with an unmatched endpoint and a blocking edge.



- otherwise $M \oplus p \succ M$.

Max size popular matchings

- Properties (1)-(3) are also sufficient for popularity.

Max size popular matchings

- Properties (1)-(3) are also sufficient for popularity.
- Suppose M also satisfies this 4th property:

Max size popular matchings

- Properties (1)-(3) are also sufficient for popularity.

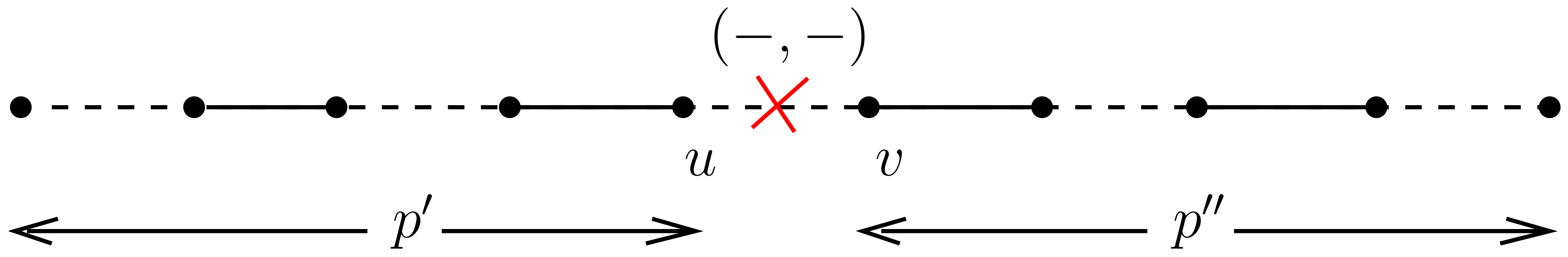
- Suppose M also satisfies this 4th property:

(4) there is no *augmenting path* wrt M in G_M.

Max size popular matchings

- Properties (1)-(3) are also sufficient for popularity.
- Suppose M also satisfies this 4th property:

(4) there is no *augmenting path* wrt M in G_M.

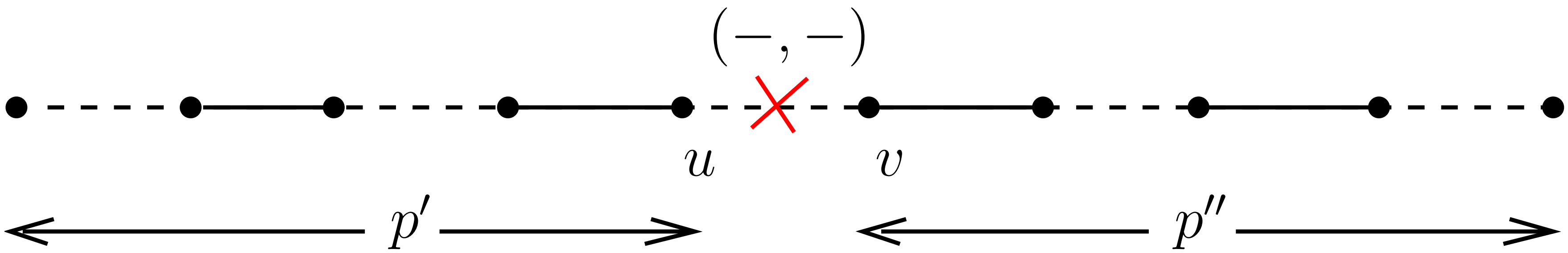


Max size popular matchings

- Properties (1)-(3) are also sufficient for popularity.

- Suppose M also satisfies this 4th property:

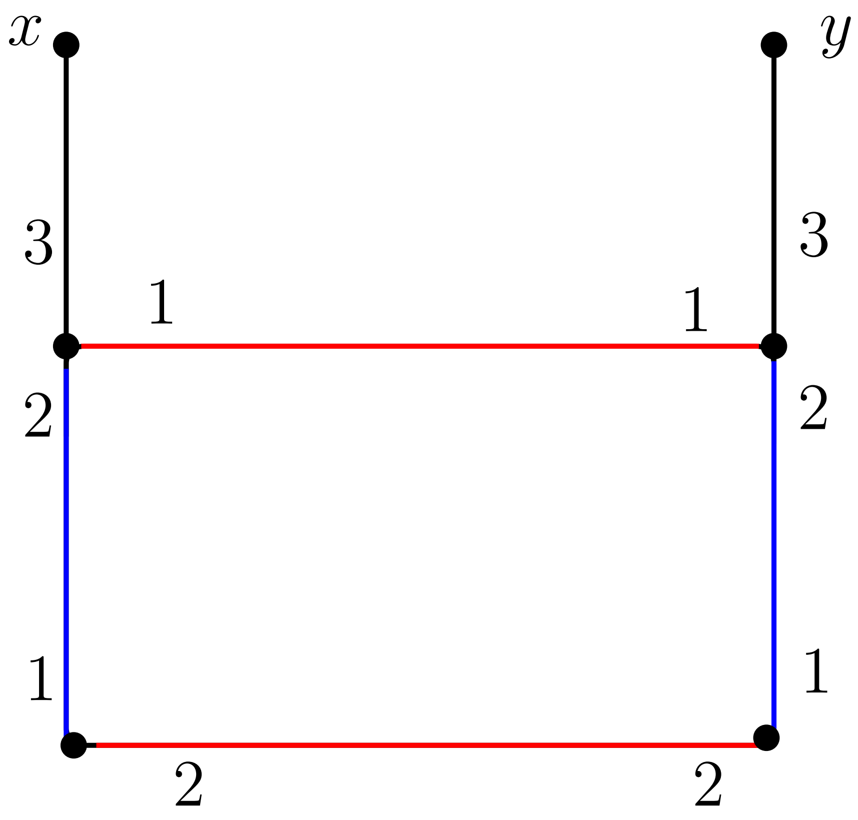
(4) there is no *augmenting path* wrt M in G_M.



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

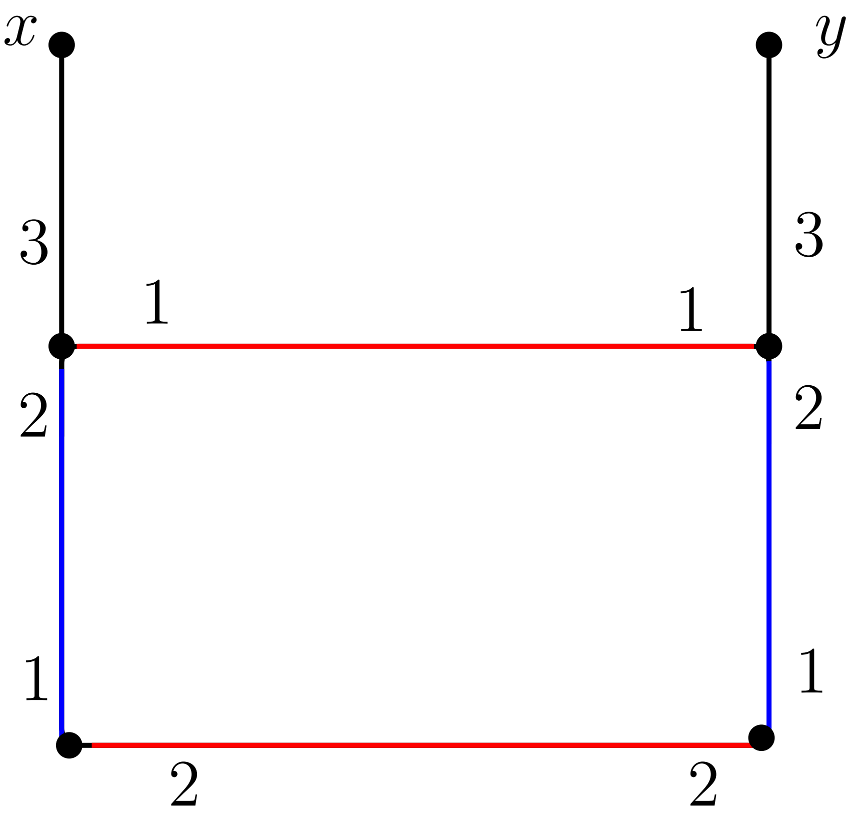
Max size popular matchings

- Property (4) is not necessary for max size popular matchings.



Max size popular matchings

- Property (4) is not necessary for max size popular matchings.



- G_M has an augmenting path wrt the red matching M.

Our first attempt

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

Our first attempt

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

Our first attempt

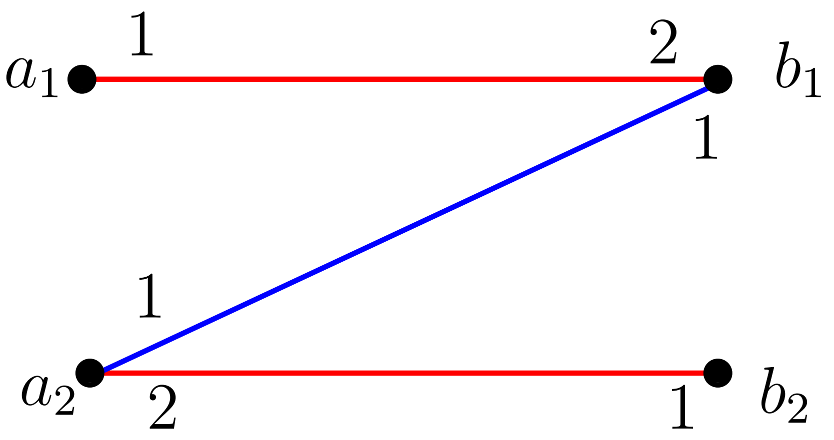
- Goal: To compute a matching that satisfies these 4 properties.
  - *Idea*: come up with a suitable partition (L, R) of $\mathcal{A} \cup \mathcal{B}$ such that
    - Gale-Shapley algorithm on (L, R) yields such a matching.

Our first attempt

- Goal: To compute a matching that satisfies these 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.
- An algorithm with running time $O(mn)$ to compute a max size popular matching in G. (Huang and K 2013)

Min vs max size popular matchings

- The blue matching is a minimum size popular matching.



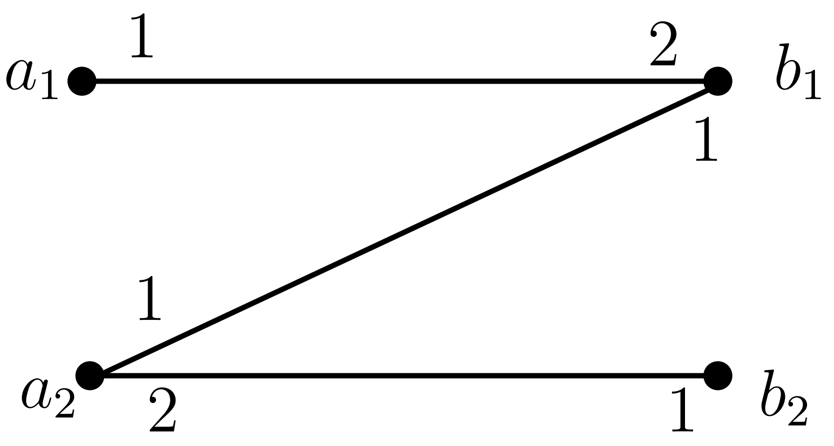
- The red matching is a maximum size popular matching.

Gale-Shapley algorithm for stable matchings

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

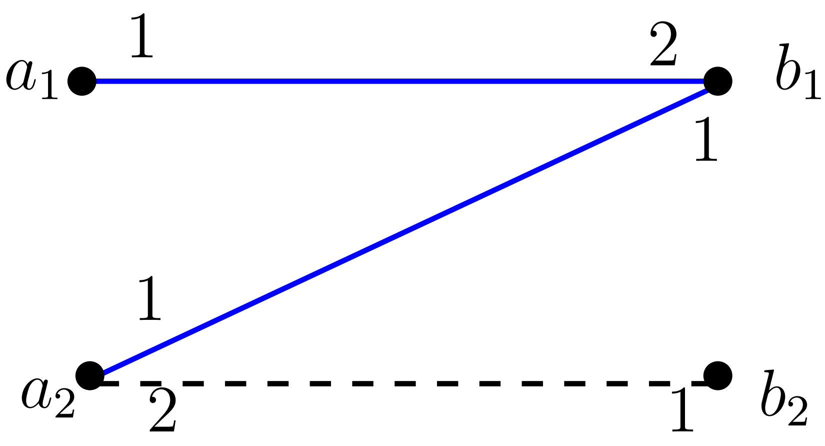
Gale-Shapley algorithm for stable matchings

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



Gale-Shapley algorithm for stable matchings

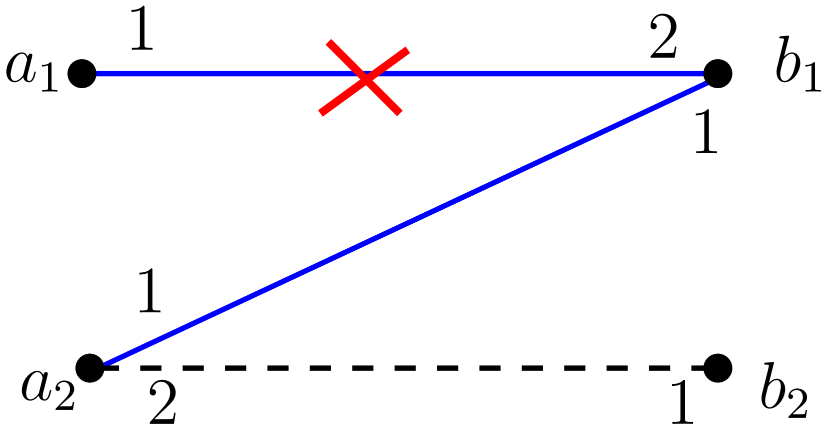
- *Men (vertices of $\mathcal{A}$) propose and Women (those in $\mathcal{B}$) dispose.*



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

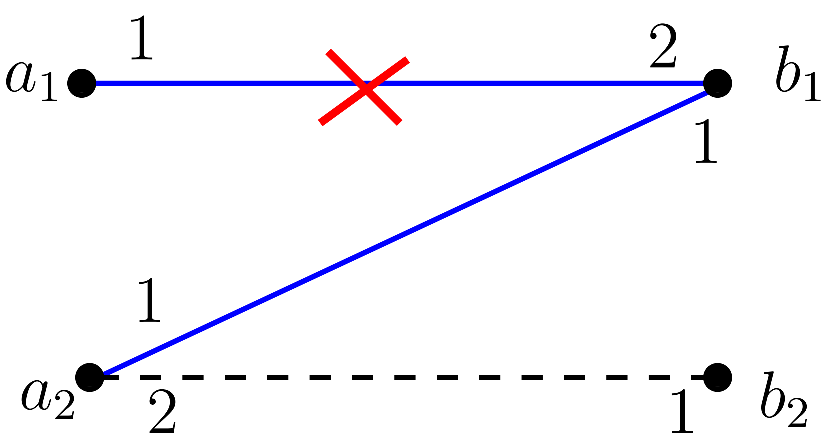
Gale-Shapley algorithm for stable matchings

- b_1 rejects a_1 and accepts a_2.



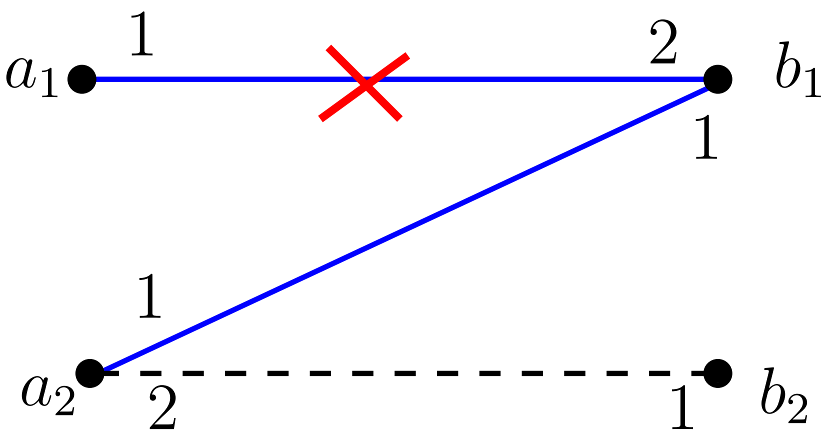
Gale-Shapley algorithm for stable matchings

- b_1 rejects a_1 and accepts a_2.

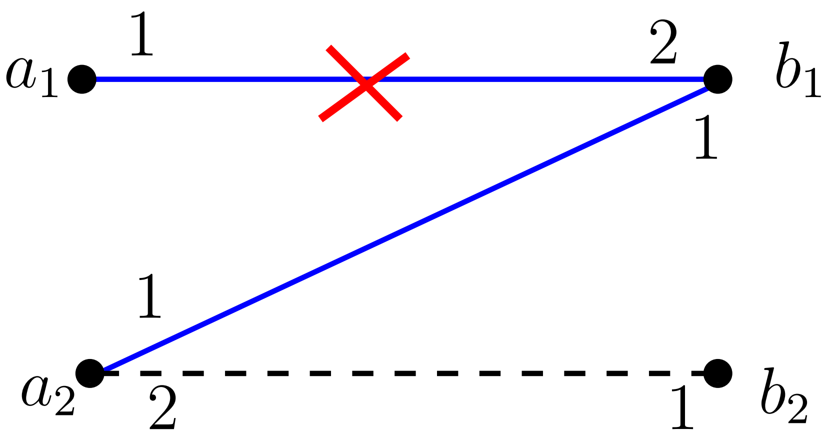


- The algorithm terminates when every man is either rejected by all his nbrs or gets matched to some nbr.

Modifying Gale-Shapley ...

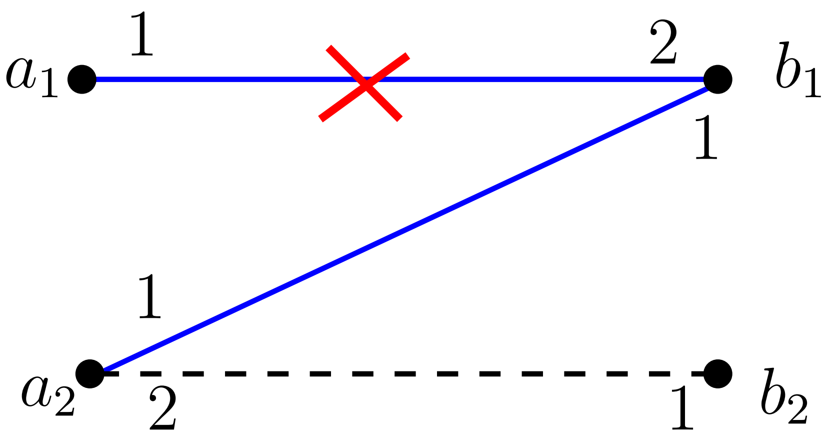


Modifying Gale-Shapley ...



- 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.

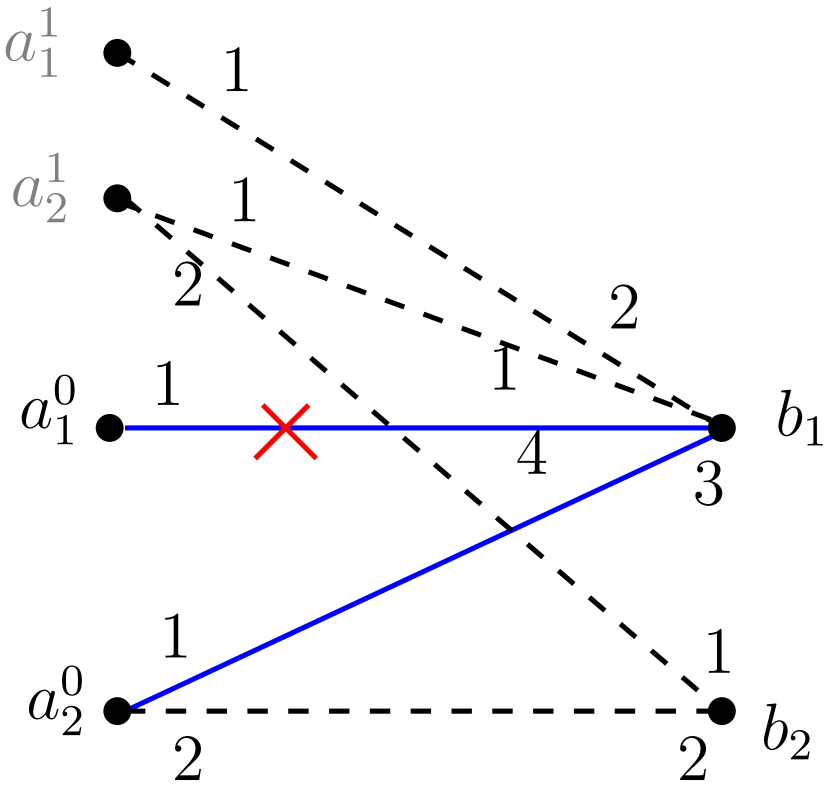
Implementing this idea

- 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 order.

Implementing this idea

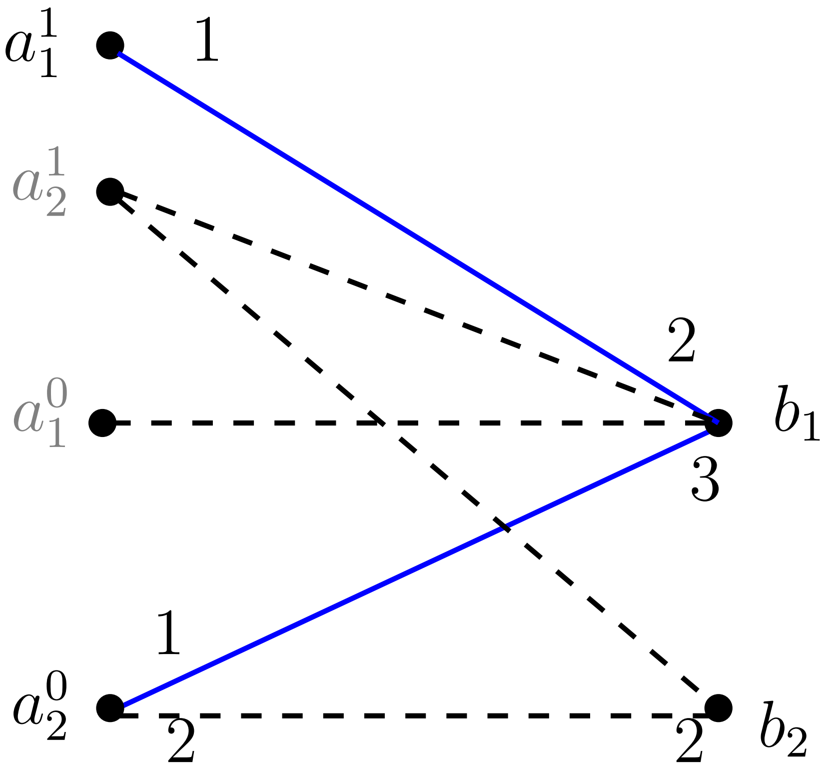
- Have *two* copies a^0 and a^1 of every man a:
  - there will be two edges (a^1, b) and (a^0, b) corresponding to every edge (a, b) in G.
    - every woman prefers a level 1 nbr to a level 0 nbr.
      - $*$ among level 1 nbrs: her original order.
      - $*$ among level 0 nbrs: her original order.

In the new graph



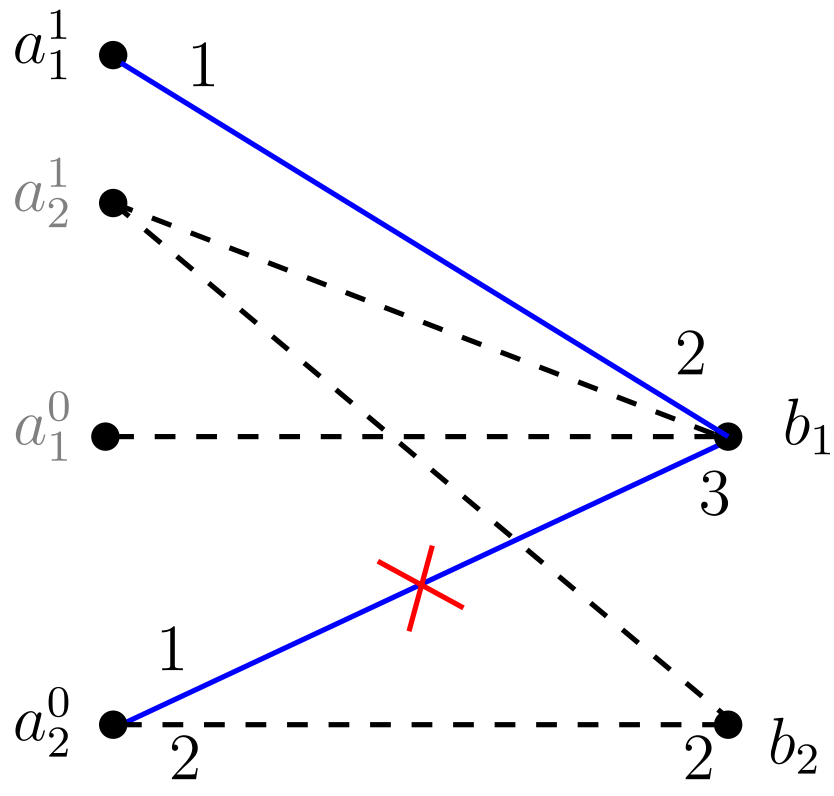
- a_1^0 is rejected by his only neighbor b_1.

In the new graph



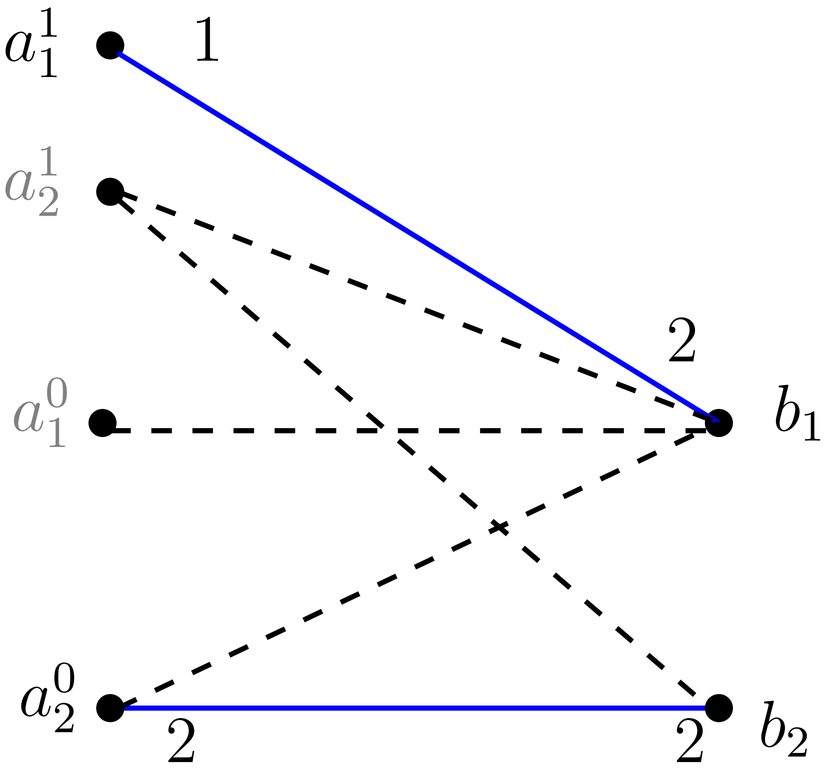
- So a_1^1 becomes active and proposes to b_1.

In the new graph



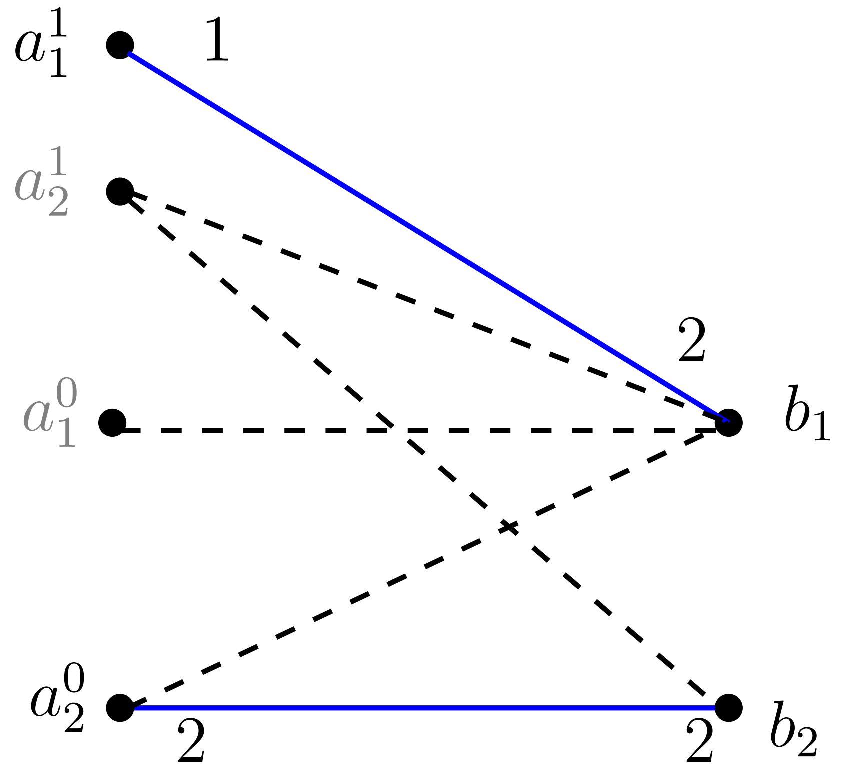
- b_1 accepts a_1^1 and rejects a_2^0.

In the new graph



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

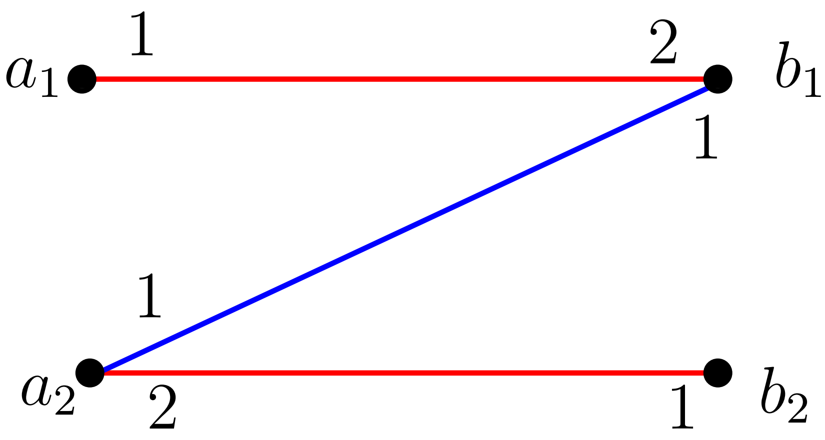
In the new graph



- 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 2014)

- Let G_2 be the graph on $A_2 \cup \mathcal{B}$ where A_2 consists of two copies a^0 and a^1 of each $a \in \mathcal{A}$.

A linear time algorithm (K 2014)

- Let G_2 be the graph on $A_2 \cup \mathcal{B}$ where 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 2014)

- Let G_2 be the graph on $A_2 \cup \mathcal{B}$ where 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 G_2.

A linear time algorithm (K 2014)

- Let G_2 be the graph on $A_2 \cup \mathcal{B}$ where 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 G_2.
  - when any a_i^0 is rejected by all his neighbors:

A linear time algorithm (K 2014)

- Let G_2 be the graph on $A_2 \cup \mathcal{B}$ where 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 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 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 G_2.

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

A linear time algorithm

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

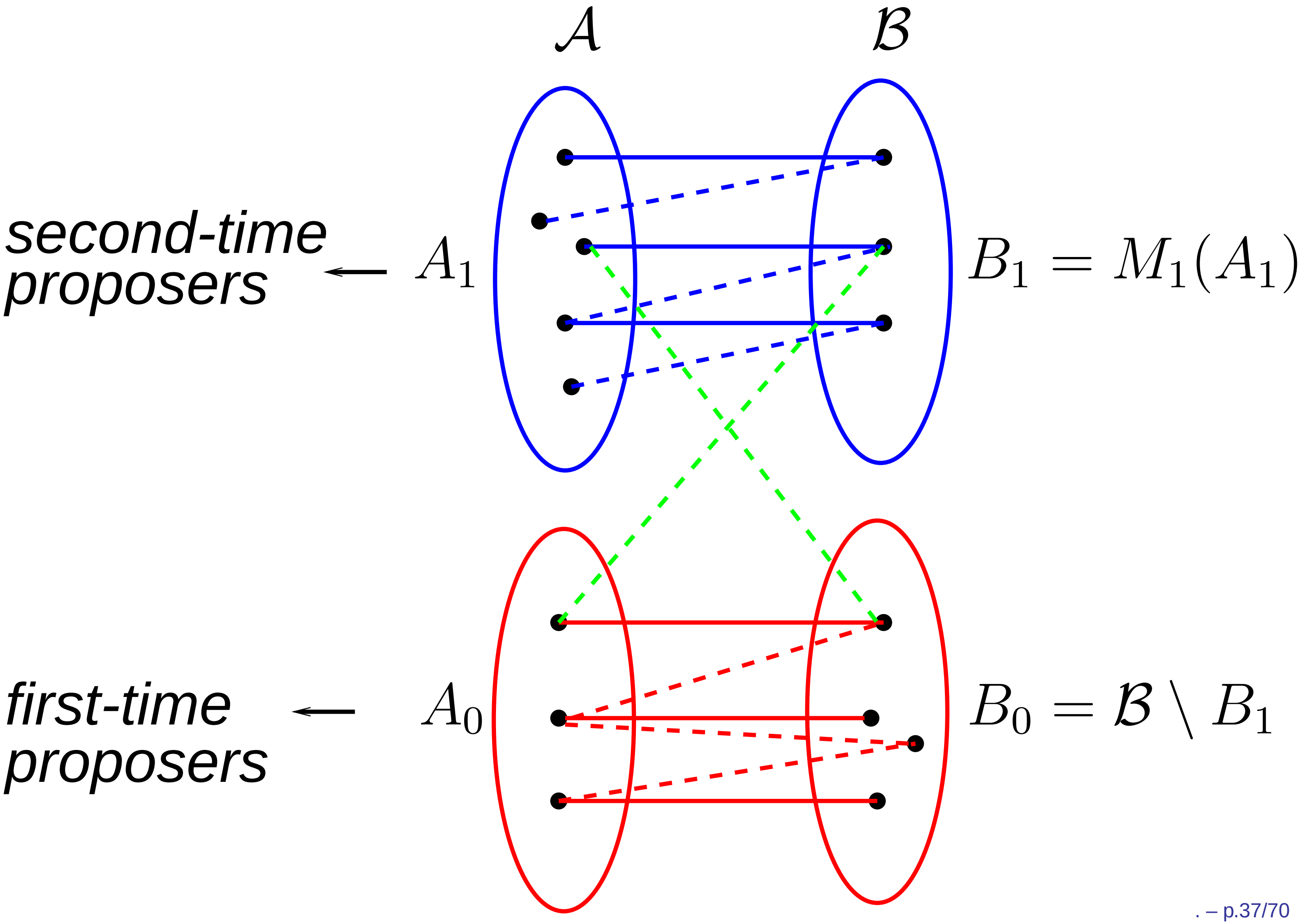
- Our algorithm is essentially Gale-Shapley algorithm on G_2.

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

- Let M_1 be the matching computed by our algorithm.

Properties of M_1

- $M_1 \subseteq (A_0 \times B_0) \cup (A_1 \times B_1)$.



Properties of M_1

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

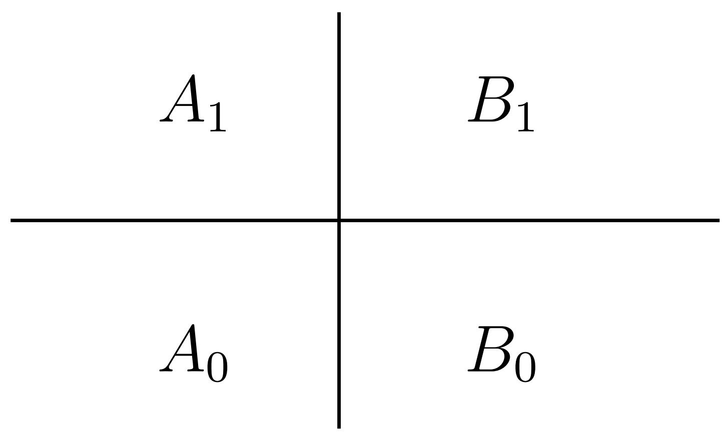
Properties of M_1

- All unmatched vertices are in $A_1 \cup B_0$.
- M_1 restricted to $A_i \cup B_i$ $(i = 0, 1)$ is stable.

A_1	B_1
A_0	B_0

Properties of M_1

- All unmatched vertices are in $A_1 \cup B_0$.
- M_1 restricted to $A_i \cup B_i$ $(i = 0, 1)$ is stable.



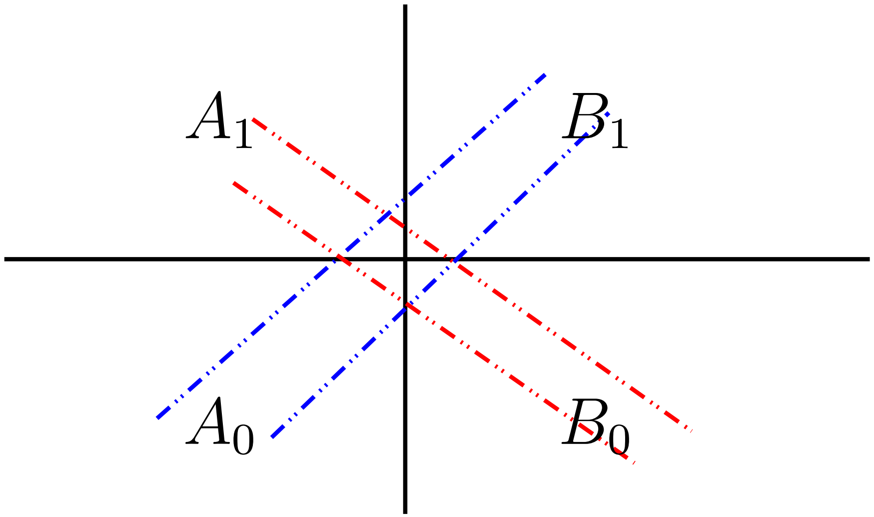
- Any blocking edge to M_1 has to be in $A_0 \times B_1$.

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

- Every edge $(a, b) \in A_1 \times B_0$ is negative wrt M_1.

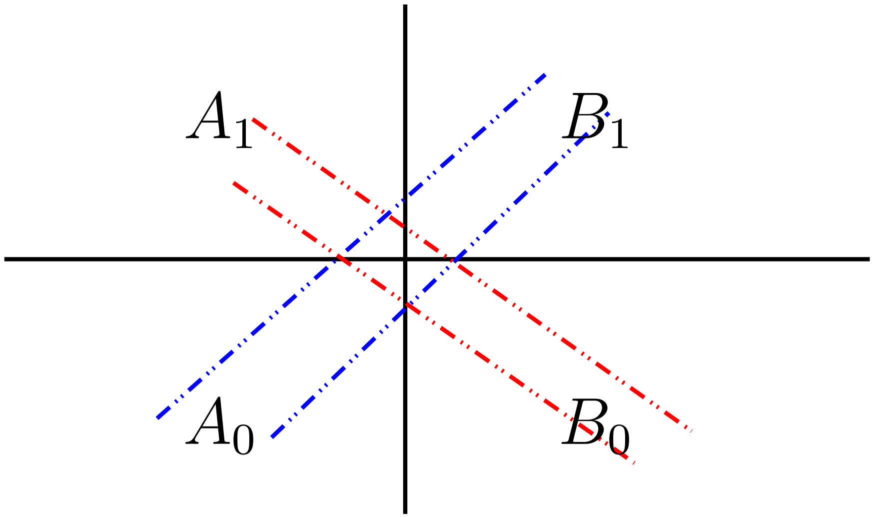
Partition of $\mathcal{A}$ and $\mathcal{B}$

- Every edge $(a, b) \in A_1 \times B_0$ is negative wrt M_1.



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

- Every edge $(a, b) \in A_1 \times B_0$ is negative wrt M_1.



- Thus G_{M_1} has no edge in $A_1 \times B_0$.

Popularity of M_1

- Consider the subgraph G_{M_1}.

Popularity of M_1

- Consider the subgraph G_{M_1}.
  - M_1 has the following properties in G_{M_1}:

Popularity of M_1

- Consider the subgraph G_{M_1}.
  - M_1 has the following properties in G_{M_1}:
    - no alternating cycle has a blocking edge.

Popularity of M_1

- Consider the subgraph G_{M_1}.
  - M_1 has the following properties in G_{M_1}:
    - no alternating cycle has a blocking edge.
    - no alternating path has 2 blocking edges.

Popularity of M_1

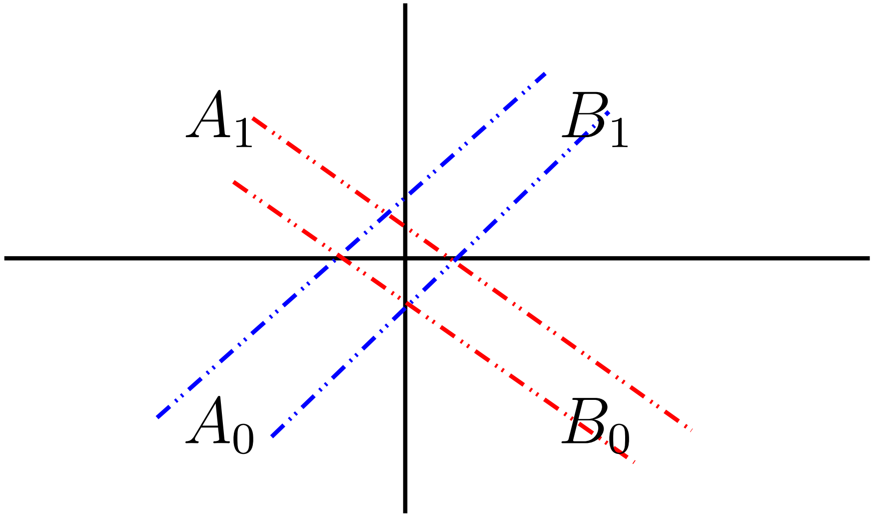
- Consider the subgraph G_{M_1}.
  - M_1 has the following properties in G_{M_1}:
    - no alternating cycle has a blocking edge.
    - no alternating path has 2 blocking edges.
    - no alternating path with an unmatched endpoint has a blocking edge.

Size of the matching M_1

- All unmatched men are in A_1 and all unmatched women are in B_0.

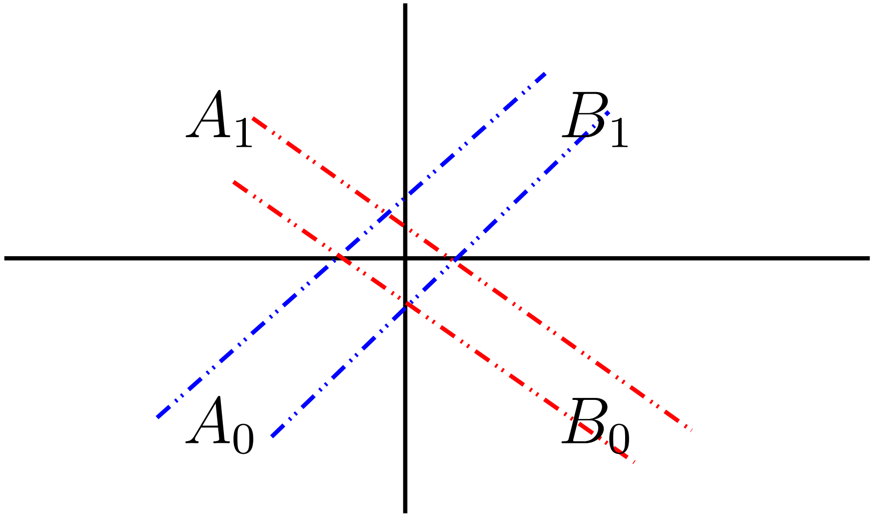
Size of the matching M_1

- All unmatched men are in A_1 and all unmatched women are in B_0.



Size of the matching M_1

- All unmatched men are in A_1 and all unmatched women are in B_0.



- There is *no* augmenting path wrt M_1 in G_{M_1}.

Size of the matching M_1

- Thus M_1 satisfies properties (1)-(4).

Size of the matching M_1

- Thus M_1 satisfies properties (1)-(4).

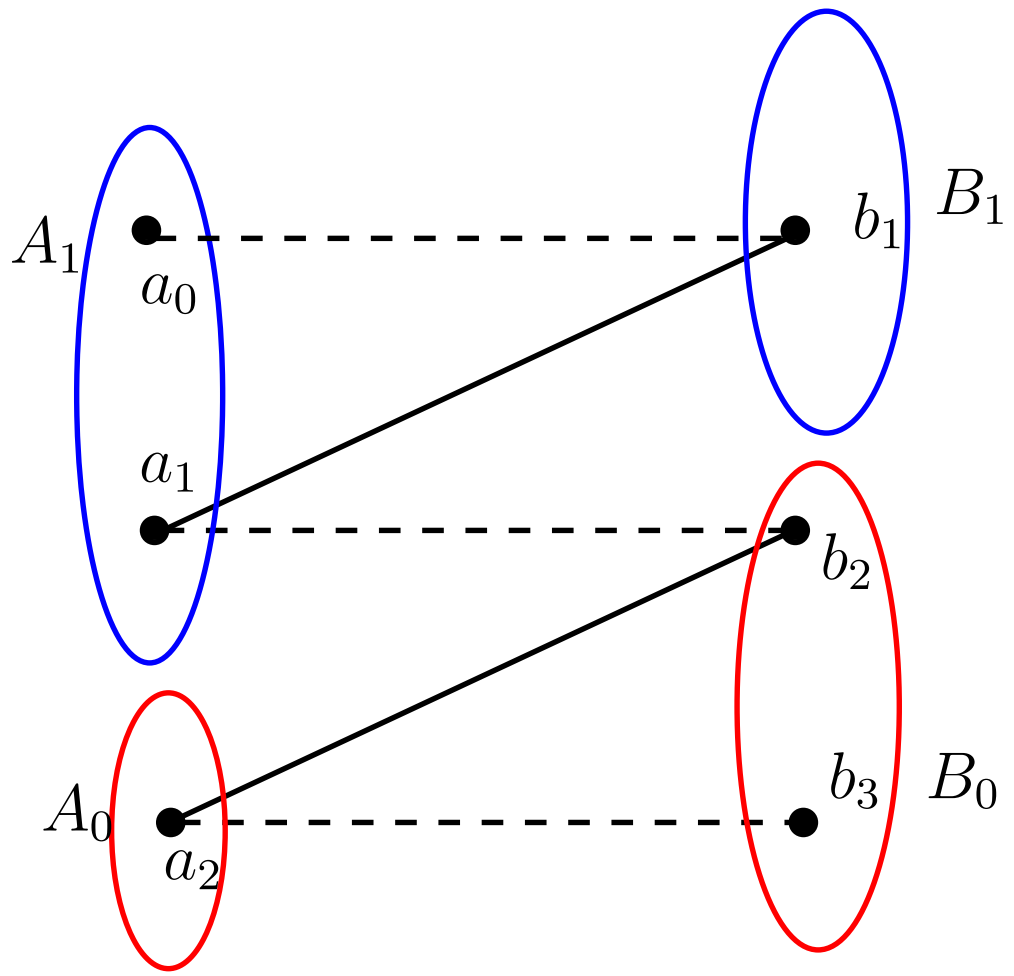
  - M_1 is a maximum size popular matching.

Size of the matching M_1

- Thus M_1 satisfies properties (1)-(4).
  - M_1 is a maximum size popular matching.
- What about $|M_1|$ in terms of $|M_{max}|$?

Size of the matching M_1

- Any augmenting path wrt M_1 in G has size ≥ 5:



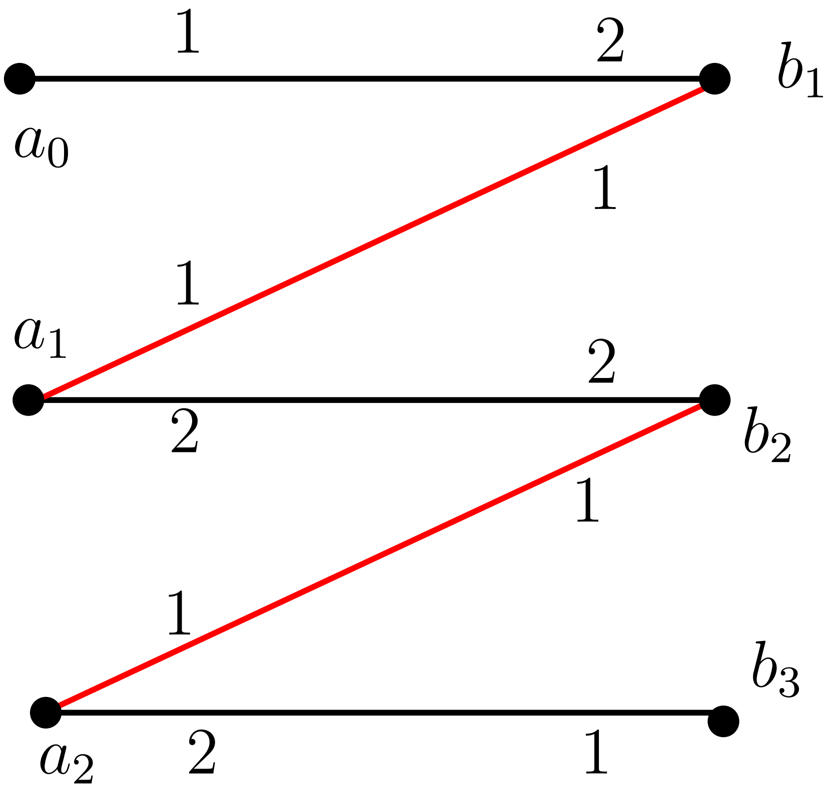
Size of the matching M_1

- Any augmenting path wrt M_1 in G has size $\geq 5 \Rightarrow$

Size of the matching M_1

- Any augmenting path wrt M_1 in G has size $\geq 5 \Rightarrow$
  - $|M_1| \geq \frac{2}{3}|M_{max}|$.

A tight example for the $2/3$ bound



- $|M_1| = 2$ while $|M_{max}| = 3$.

Larger size matchings

- Trade-off between popularity and size?

Larger size matchings

- Trade-off between popularity and size?
- Unpopularity factor $u(\cdot)$ (McCutchen 2008)

Larger size matchings

- Trade-off between popularity and size?
- Unpopularity factor $u(\cdot)$ (McCutchen 2008)
  - 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 $u(\cdot)$ (McCutchen 2008)
  - 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 $u(\cdot)$ (McCutchen 2008)
  - 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 $\Leftrightarrow u(M) \leq 1$.
  - there is a matching M_1 with $u(M_1) \leq 1$ and $|M_1| \geq \frac{2}{3}|M_{max}|$.

Popularity vs Size

- M is popular $\Leftrightarrow u(M) \leq 1$.
  - there is a matching M_1 with $u(M_1) \leq 1$ and $|M_1| \geq \frac{2}{3}|M_{max}|$.
  - for any integer $k > 2$, is there a matching M_{k-1} with $u(M_{k-1}) \leq k - 1$ and $|M_{k-1}| \geq \frac{k}{k+1}|M_{max}|$?

Popularity vs Size

- M is popular $\Leftrightarrow u(M) \leq 1$.
  - there is a matching M_1 with $u(M_1) \leq 1$ and $|M_1| \geq \frac{2}{3}|M_{max}|$.
  - for any integer $k > 2$, is there a matching M_{k-1} with $u(M_{k-1}) \leq k-1$ and $|M_{k-1}| \geq \frac{k}{k+1}|M_{max}|$?
- Is there an $M^* \equiv$ a maximum cardinality matching s.t. for each maximum cardinality matching M: $M^* \succeq M$?

Extending the 2-level algorithm

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

Extending the 2-level algorithm

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

Extending the 2-level algorithm

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

  - so the graph becomes G_k on $\mathcal{A}_k \cup \mathcal{B}$.

    - $\mathcal{A}_k$ 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 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 G_k.
- In 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 G_k.
- In 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 G_k.

- In 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 G_k.

- In 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

- Initially only level 0 men are active.

The k-level algorithm

- Initially only level 0 men are active.
- Essentially Gale-Shapley with the active men proposing and women disposing:

The k-level algorithm

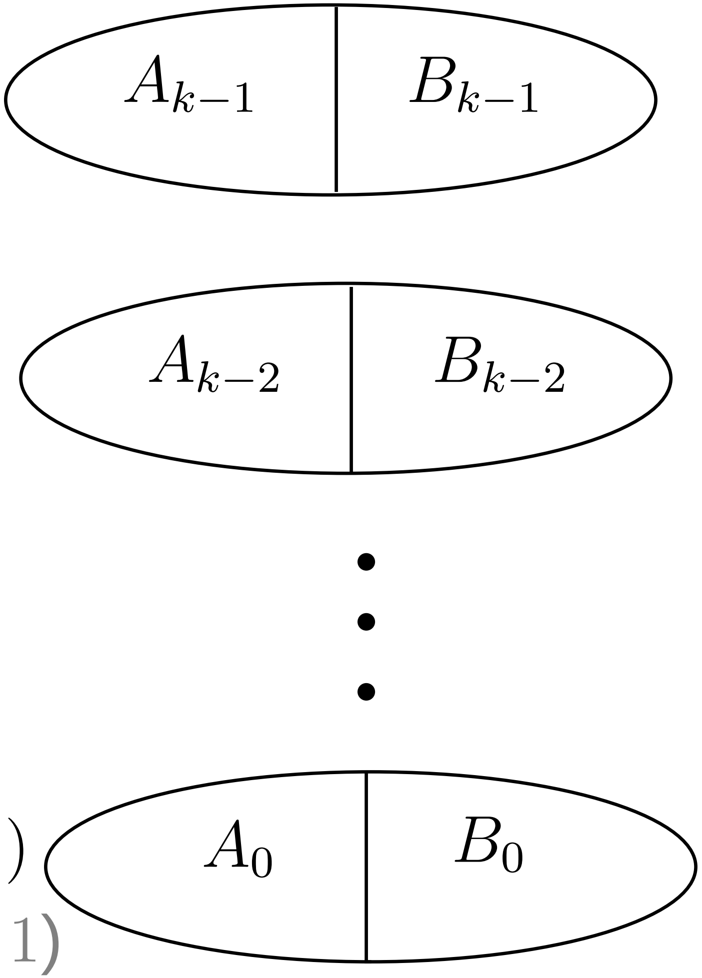
- Initially only level 0 men are active.
- Essentially Gale-Shapley with the active men proposing and women disposing:
  - $i < k - 1$: if a^i is rejected by all his neighbors, then a^{i+1} becomes active.

The k-level algorithm

- Initially only level 0 men are active.
- 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 M_{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$\}$.

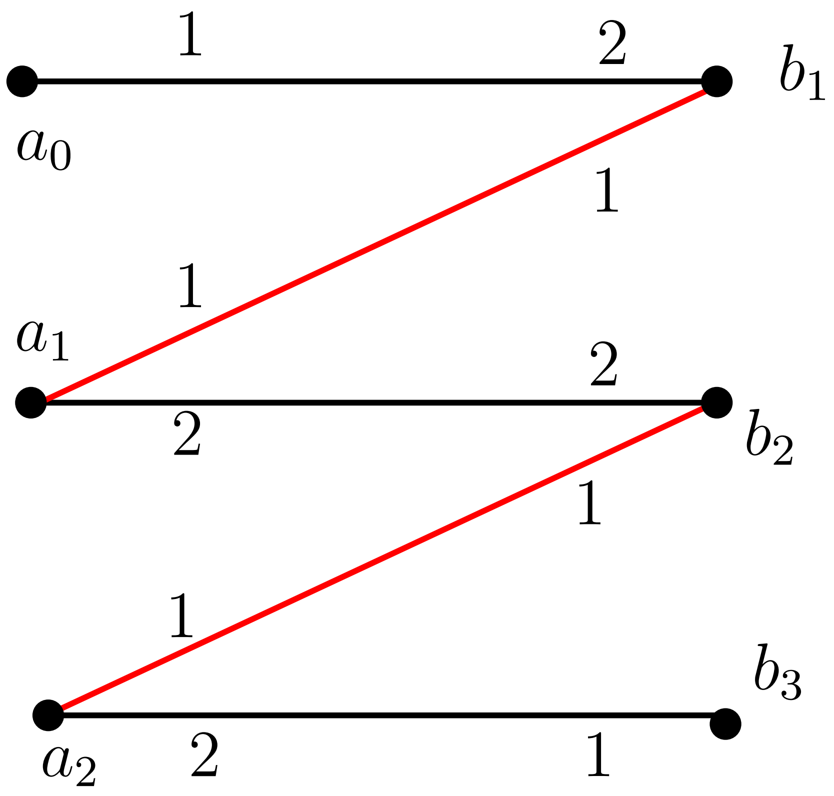


- $B_i = M_{k-1}(A_i)$

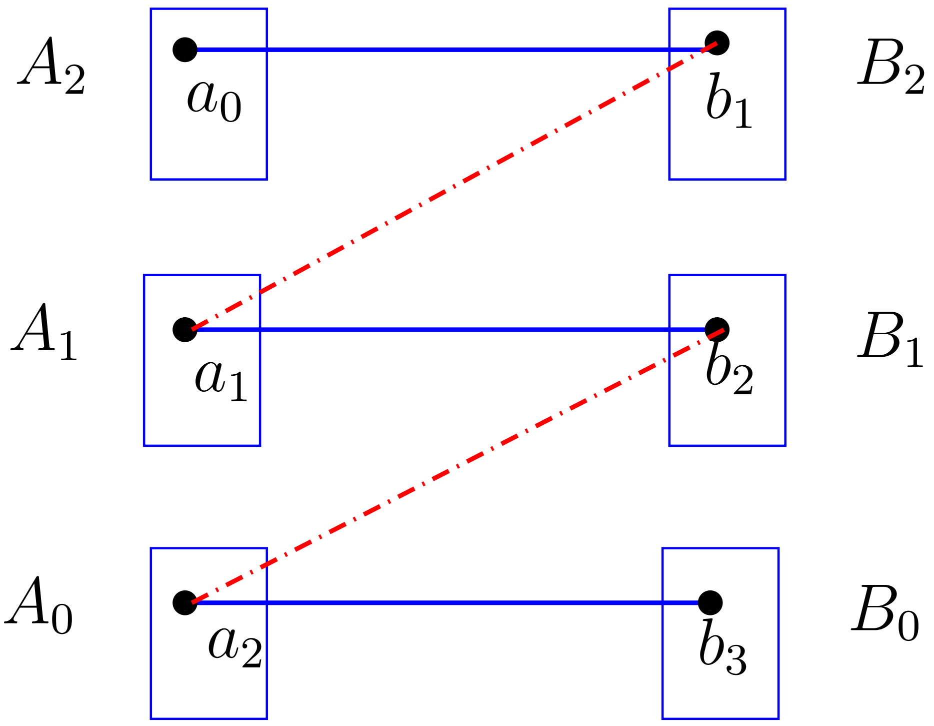
(for $1 \leq i \leq k-1$)

The 3-level algorithm

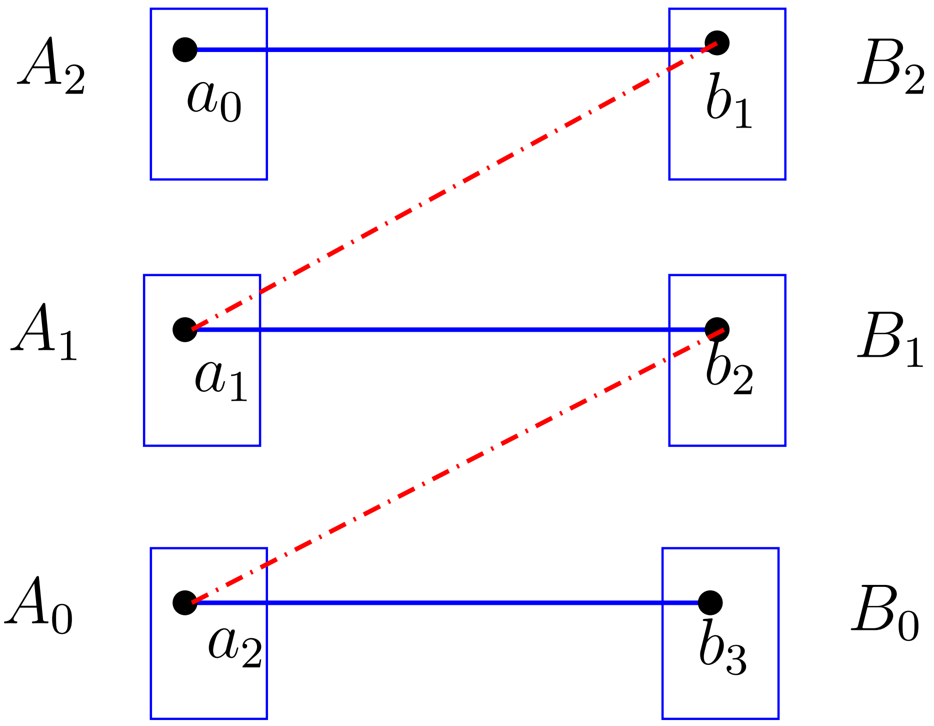
- Say we run the 3-level algorithm on our tight example for the 2-level algorithm ...



In the 3-level algorithm



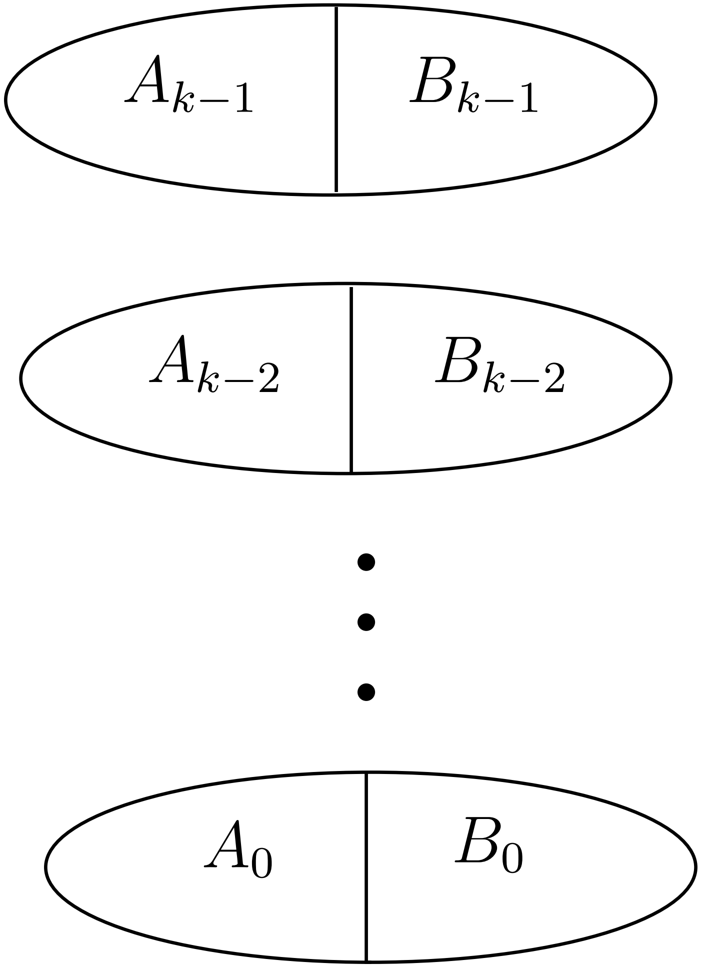
In the 3-level algorithm



- The matching $M_2 = \{(a_0, b_1), (a_1, b_2), (a_2, b_3)\}$ is computed by the 3-level algorithm.

Properties of the matching M_{k-1}

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



Properties of the matching M_{k-1}

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

Properties of the matching M_{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 M_{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 M_{k-1} has length $\geq 2k+1$.

Properties of the matching M_{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 M_{k-1} has length $\geq 2k+1$.
    - hence $|M_{k-1}| \geq \frac{k}{k+1} |M_{max}|$.

Unpopularity of M_{k-1}

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

Unpopularity of M_{k-1}

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

Unpopularity of M_{k-1}

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

Unpopularity of M_{k-1}

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

Unpopularity of M_{k-1}

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

Trade-off between size and unpopularity

- This means $u(M_{k-1}) \leq k - 1$.

Trade-off between size and unpopularity

- This means $u(M_{k-1}) \leq k - 1$.

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

Trade-off between size and unpopularity

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

Trade-off between size and unpopularity

- This means $u(M_{k-1}) \leq k-1$.
- Thus for any $k \geq 2$, there exists a matching M_{k-1} s.t. $u(M_{k-1}) \leq k-1$ and $|M_{k-1}| \geq \frac{k}{k+1}|M_{max}|$.
- Also for any matching M:
  - $|M| \geq |M_{k-1}| \implies M_{k-1} \succeq M$.

Trade-off between size and unpopularity

- This means $u(M_{k-1}) \leq k-1$.
- Thus for any $k \geq 2$, there exists a matching M_{k-1} s.t. $u(M_{k-1}) \leq k-1$ and $|M_{k-1}| \geq \frac{k}{k+1}|M_{max}|$.
- Also for any matching M:
  - $|M| \geq |M_{k-1}| \implies M_{k-1} \succeq M$.
    - in particular, $|M| > |M_{k-1}| \implies M_{k-1} \succ M$.

The boundary cases

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

The boundary cases

- M_{k-1} can be computed in $O(mk)$ time.
- $k = 2$: M_1 is a maximum size popular matching

The boundary cases

- M_{k-1} can be computed in $O(mk)$ time.
- $k = 2$: M_1 is a maximum size popular matching
- $k = n_0$: let M^* denote $M_{(n_0-1)}$.

The boundary cases

- M_{k-1} can be computed in $O(mk)$ time.
- $k = 2$: M_1 is a maximum size popular matching
- $k = n_0$: let M^* denote $M_{(n_0-1)}$.
  - $|M^*| \geq \frac{n_0}{n_0+1}|M_{max}|$, so $|M^*| = |M_{max}|$ (since $|M_{max}| \leq n_0$).

The boundary cases

- M_{k-1} can be computed in $O(mk)$ time.
- $k = 2$: M_1 is a maximum size popular matching
- $k = n_0$: let M^* denote $M_{(n_0-1)}$.
  - $|M^*| \geq \frac{n_0}{n_0+1}|M_{max}|$, so $|M^*| = |M_{max}|$ (since $|M_{max}| \leq n_0$).
  - for any max cardinality matching M: $M^* \succeq M$.

With ties in preference lists

- Input $G = (\mathcal{A} \cup \mathcal{B}, E)$: ties allowed in preference lists.

With ties in preference lists

- Input $G = (\mathcal{A} \cup \mathcal{B}, E)$: ties allowed in preference lists.

  - popular matchings need not exist in G.

With ties in preference lists

- Input $G = (\mathcal{A} \cup \mathcal{B}, E)$: ties allowed in preference lists.
  - popular matchings need not exist in G.
    - determining if one exists is NP-hard.
      (Biró, Irving, and Manlove 2010)

With ties in preference lists

- Input $G = (\mathcal{A} \cup \mathcal{B}, E)$: ties allowed in preference lists.
  - popular matchings need not exist in G.
    - determining if one exists is NP-hard. (Biró, Irving, and Manlove 2010)
- The problem is NP-hard even with one-sided ties. (Cseh, Huang, and K 2015)

One-sided ties

- In fact, the popular matching problem is NP-hard in $G = (\mathcal{A} \cup \mathcal{B}, E)$ where:

One-sided ties

- In fact, the popular matching problem is NP-hard in $G = (\mathcal{A} \cup \mathcal{B}, E)$ where:
  - each $a \in \mathcal{A}$ has a strict preference list

One-sided ties

- In fact, the popular matching problem is NP-hard in $G = (\mathcal{A} \cup \mathcal{B}, E)$ where:
  - each $a \in \mathcal{A}$ has a strict preference list
  - each $b \in \mathcal{B}$ either has a strict preference list or puts all neighbors into a single tie.

One-sided ties

- In fact, the popular matching problem is NP-hard in $G = (\mathcal{A} \cup \mathcal{B}, E)$ where:
  - each $a \in \mathcal{A}$ has a strict preference list
  - each $b \in \mathcal{B}$ either has a strict preference list or puts all neighbors into a single tie.
    - the case where each $b \in \mathcal{B}$ puts all neighbors into a single tie has an $O(n^2)$ algorithm.

One-sided preference lists

- The model of *one-sided* preference lists:
  $G = (\mathcal{A} \cup \mathcal{B}, E)$

One-sided preference lists

- The model of *one-sided* preference lists: $G = (\mathcal{A} \cup \mathcal{B}, E)$
  - $\mathcal{A}$: a set of applicants; $\mathcal{B}$: a set of training posts.

One-sided preference lists

- The model of *one-sided* preference lists:
  $G = (\mathcal{A} \cup \mathcal{B}, E)$
  - $\mathcal{A}$: a set of applicants; $\mathcal{B}$: a set of training posts.
  - each applicant has a preference list; posts have no preferences.

One-sided preference lists

- The model of *one-sided* preference lists: $G = (\mathcal{A} \cup \mathcal{B}, E)$
  - $\mathcal{A}$: a set of applicants; $\mathcal{B}$: a set of training posts.
  - each applicant has a preference list; posts have no preferences.
    - only applicants vote while comparing 2 matchings

One-sided preference lists

- The model of *one-sided* preference lists:
  $G = (\mathcal{A} \cup \mathcal{B}, E)$
  - $\mathcal{A}$: a set of applicants; $\mathcal{B}$: a set of training posts.
  - each applicant has a preference list; posts have no preferences.
    - only applicants vote while comparing 2 matchings
- Determine if G admits a popular matching.

One-sided preference lists

- G admits a popular matching $\iff$ G' has an $\mathcal{A}$-perfect matching. (Abraham, Irving, K, and Mehlhorn 2007)

One-sided preference lists

- G admits a popular matching $\iff$ G' has an $\mathcal{A}$-perfect matching. (Abraham, Irving, K, and Mehlhorn 2007)

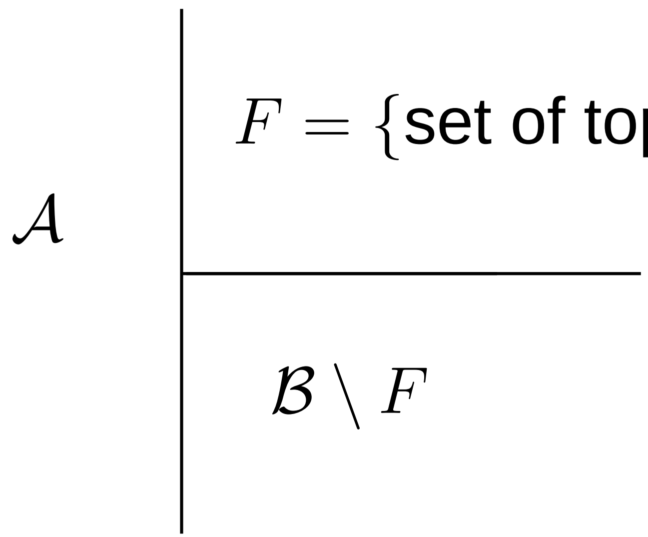
$\mathcal{A}$

$F = \{$set of top posts$\}$

$\mathcal{B} \setminus F$

One-sided preference lists

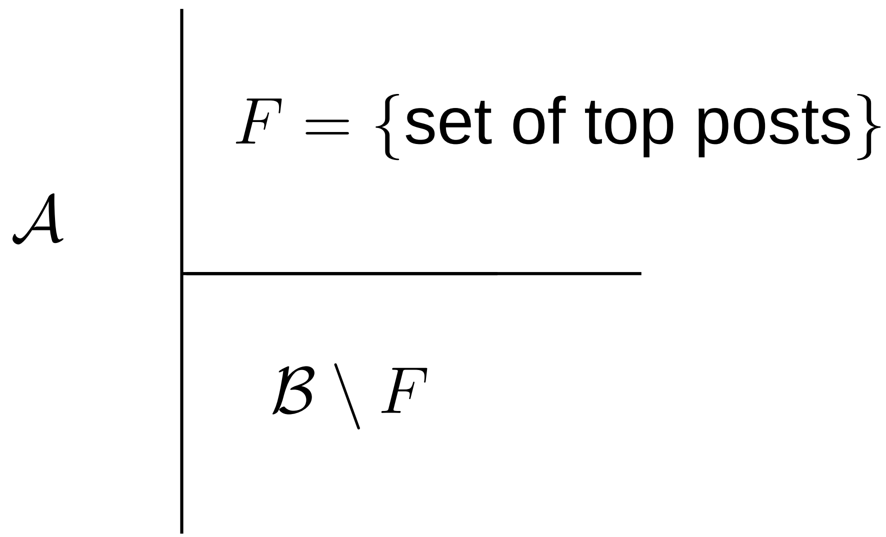
- G admits a popular matching $\iff$ G' has an $\mathcal{A}$-perfect matching. (Abraham, Irving, K, and Mehlhorn 2007)



- every $a \in \mathcal{A}$ has degree 2 in G'

One-sided preference lists

- G admits a popular matching $\iff$ G' has an $\mathcal{A}$-perfect matching. (Abraham, Irving, K, and Mehlhorn 2007)



  - every $a \in \mathcal{A}$ has degree 2 in G'

- A linear time to solve the popular matching problem: extends to the case with ties in preference lists.

Two-sided preference lists

- $G = (\mathcal{A} \cup \mathcal{B}, E)$: here each $b \in \mathcal{B}$ cares to be matched; b's neighbors are in a single tie

Two-sided preference lists

- $G = (\mathcal{A} \cup \mathcal{B}, E)$: here each $b \in \mathcal{B}$ cares to be matched; b's neighbors are in a single tie

  - here we partition $\mathcal{B}$ into *three* sets:

Two-sided preference lists

- $G = (\mathcal{A} \cup \mathcal{B}, E)$: here each $b \in \mathcal{B}$ cares to be matched; b's neighbors are in a single tie

  - here we partition $\mathcal{B}$ into *three* sets:

    - $X \subseteq F = \{$set of top posts$\}$

Two-sided preference lists

- $G = (\mathcal{A} \cup \mathcal{B}, E)$: here each $b \in \mathcal{B}$ cares to be matched; b's neighbors are in a single tie
  - here we partition $\mathcal{B}$ into *three* sets:
    - $X \subseteq F = \{$set of top posts$\}$
    - $Y \subseteq F \cup S$, where $S = \{$posts outside F that get matched in every maximum size matching in $G'\}$

Two-sided preference lists

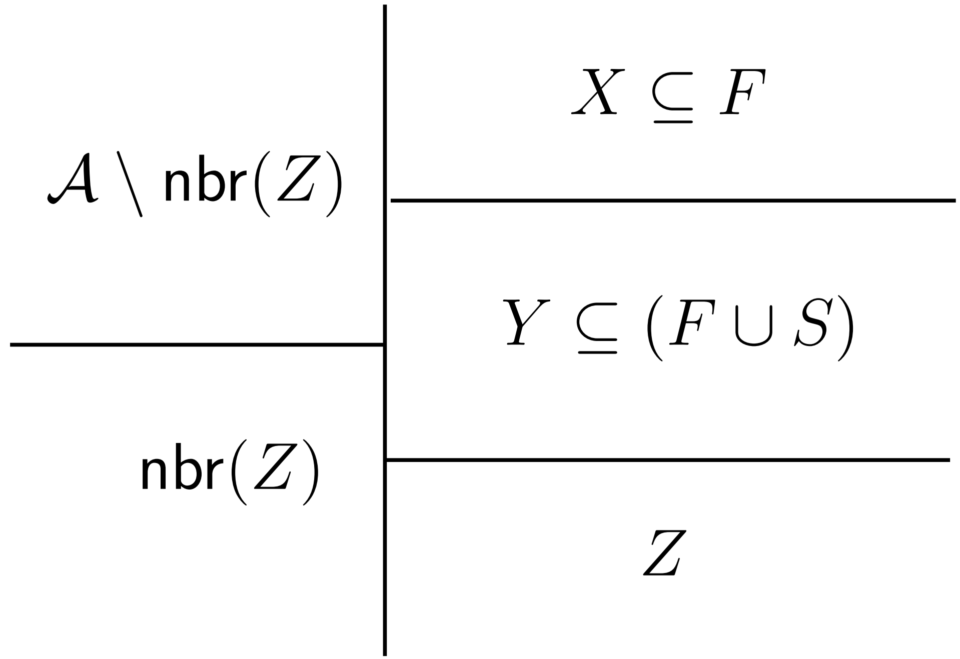
- $G = (\mathcal{A} \cup \mathcal{B}, E)$: here each $b \in \mathcal{B}$ cares to be matched; b's neighbors are in a single tie
  - here we partition $\mathcal{B}$ into *three* sets:
    - $X \subseteq F = \{$set of top posts$\}$
    - $Y \subseteq F \cup S$, where $S = \{$posts outside F that get matched in every maximum size matching in $G'\}$
    - $Z = \mathcal{B} \setminus (X \cup Y)$.

Two-sided preference lists

- The sets X, Y, and Z are constructed over n iterations.

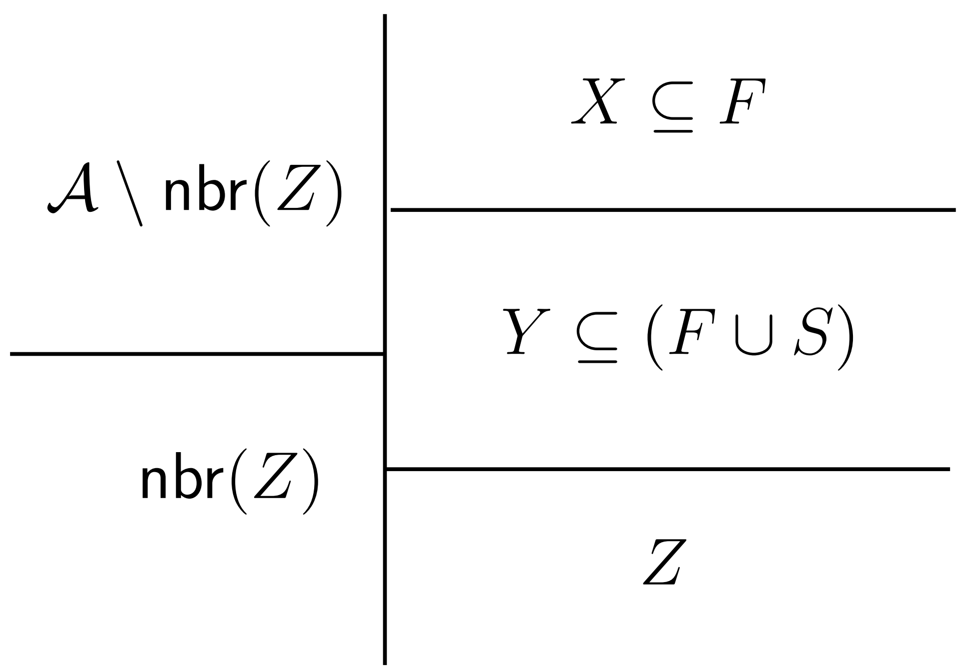
Two-sided preference lists

- The sets X, Y, and Z are constructed over n iterations.



Two-sided preference lists

- The sets X, Y, and Z are constructed over n iterations.



- G admits a popular matching $\iff$ H has an $\mathcal{A}$-perfect matching.

In general graphs

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

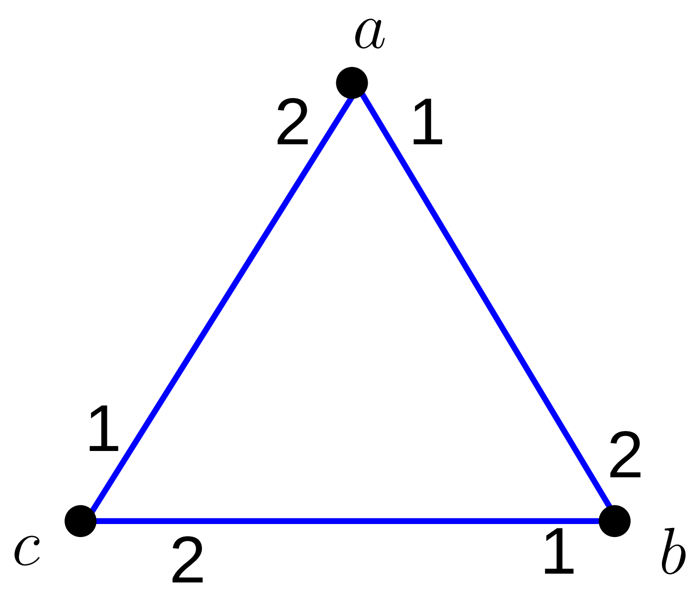
In general graphs

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

- Stable matchings need not always exist here.

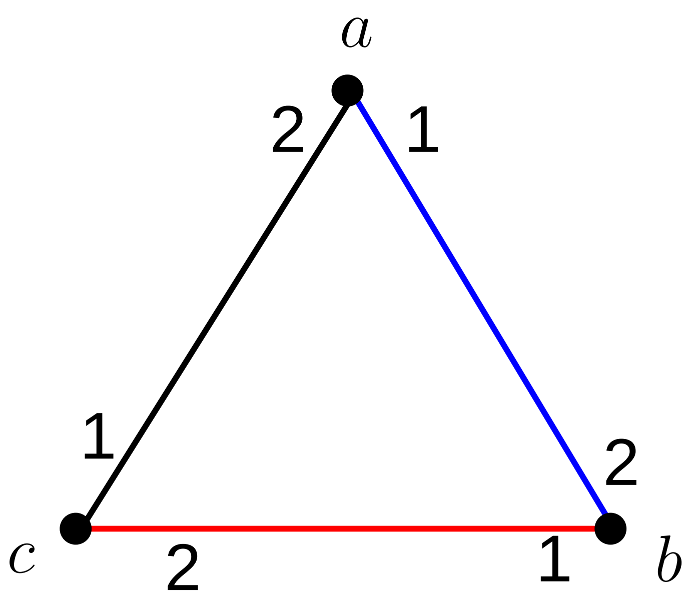
In general graphs

- Input $G = (V, E)$: a general graph with strict 2-sided preference lists
- Stable matchings need not always exist here.



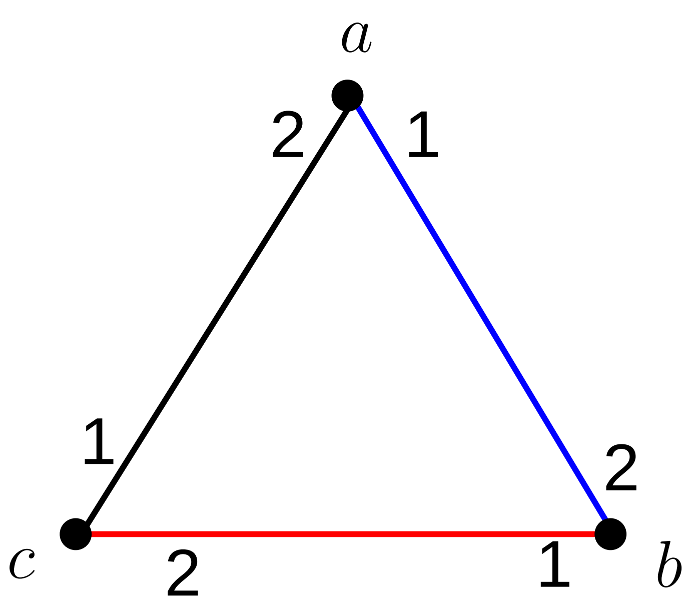
In general graphs

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



In general graphs

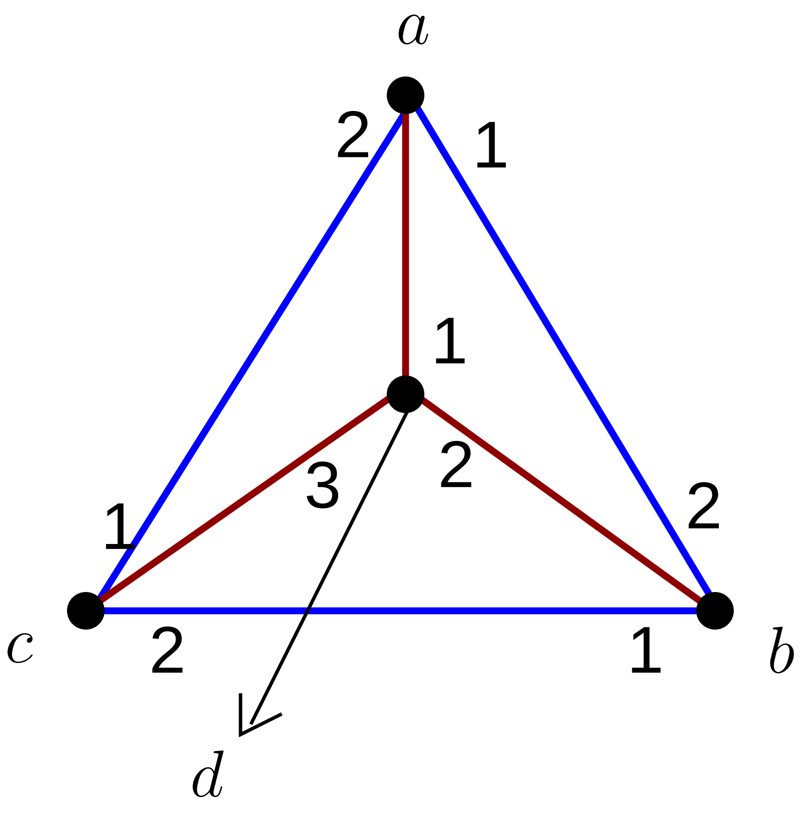
- In fact, this instance has no popular matching either.



- 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

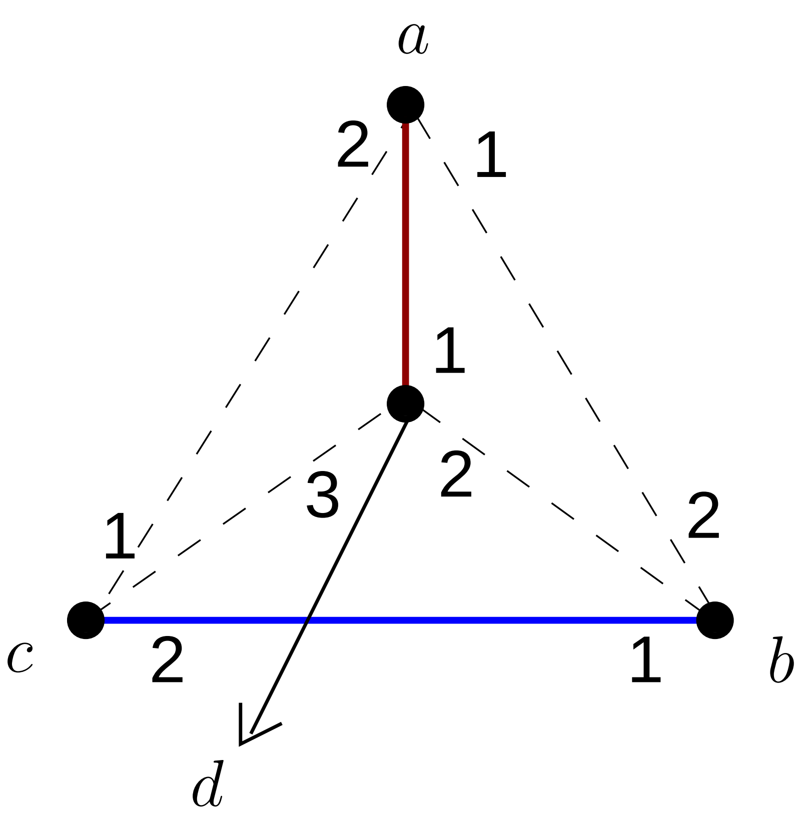
- An instance with no stable matching but with popular matchings:



- 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

- There is always a matching M in G such that $u(M)$ is $O(\log n)$. (Huang and K 2013)

In general graphs

- There is always a matching M in G such that $u(M)$ is $O(\log n)$. (Huang and K 2013)

- Computing a least unpopularity factor matching in G is NP-hard.

In general graphs

- There is always a matching M in G such that $u(M)$ is $O(\log n)$. (Huang and K 2013)
- Computing a least unpopularity factor matching in G is NP-hard.
- *Open problem*: the complexity of the popular matching problem in G.

Thank you!