Classified Matchings under one sided preferences

Meghana Nasre
IIT Madras

Recent Trends in Algorithms

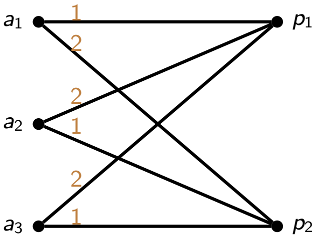
NISER, Bhubaneshwar
Feb 07, 2019

joint work with Prajakta Nimbhorkar (CMI) and Nada Pulath (IIT-M)

Problem setup

Input:

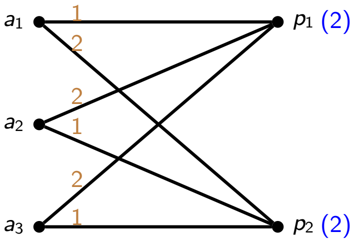
- A set of applicants A.
- A set of posts P.
- Applicants have preferences over a subset in P.



Problem setup

Input:

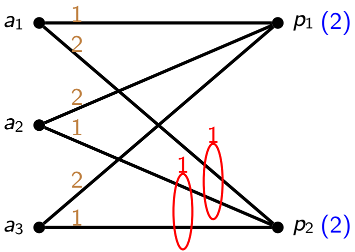
- A set of applicants A.
- A set of posts P.
- Applicants have preferences over a subset in P.
- Posts have quotas.



Problem setup

Input:

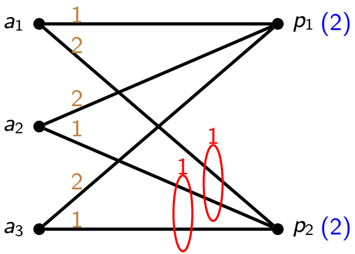
- A set of applicants A.
- A set of posts P.
- Applicants have preferences over a subset in P.
- Posts have quotas.
- Posts have classes.



Problem setup

Input:

- A set of applicants A.
- A set of posts P.
- Applicants have preferences over a subset in P.
- Posts have quotas.
- Posts have classes.

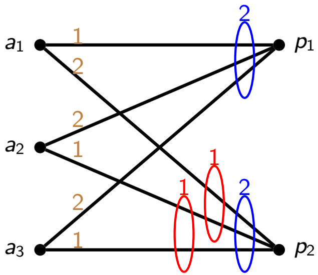


Classes are subsets on the neighborhood.

Problem setup

Input:

- A set of applicants A.
- A set of posts P.
- Applicants have preferences over a subset in P.
- Posts have quotas.
- Posts have classes.

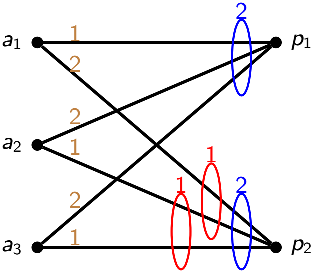


The neighborhood is a trivial class!

Problem setup

Input:

- A set of applicants A.
- A set of posts P.
- Applicants have preferences over a subset in P.
- Posts have quotas.
- Posts have classes.



Goal: Match applicants to posts optimally.

Why classifications?

Some natural constraints that can be modelled:

Allotting courses to students

- Course - may not want many students from the same Dept.

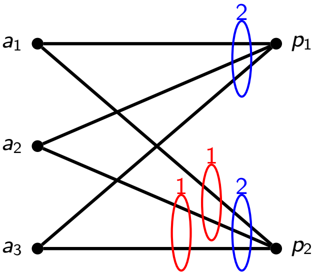
Allotting tasks to employees

- Task - wasteful to have many employees with the same skills.

Problem Setup

Input:

- A set of applicants A.
- A set of posts P.
- ~~Applicants have preferences over a subset in P.~~
- Posts have quotas.
- Posts have classes.

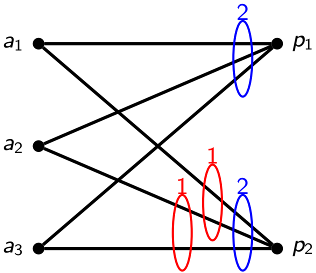


Goal: Compute a maximum cardinality matching.

Problem Setup

Input:

- A set of applicants A.
- A set of posts P.
- ~~Applicants have preferences over a subset in P.~~
- Posts have quotas.
- Posts have classes.



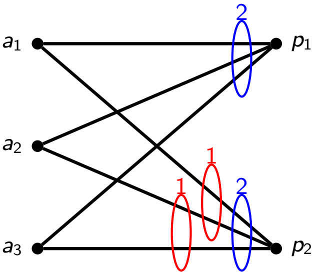
Goal: Compute a maximum cardinality matching.

- Arbitrary classes then problem is NP-Hard.

Problem Setup

Input:

- A set of applicants A.
- A set of posts P.
- ~~Applicants have preferences over a subset in P.~~
- Posts have quotas.
- Posts have classes.



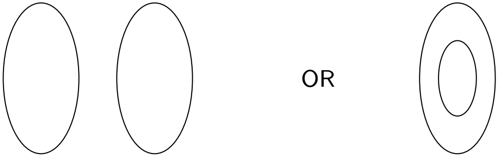
Goal: Compute a maximum cardinality matching.

- Arbitrary classes then problem is NP-Hard.
- Consider special classification families.

Laminar classification

Huang (2010); 2-sided pref.

$\boxed{\text{Laminar classification} \iff \text{any pair of classes is non-intersecting}}$

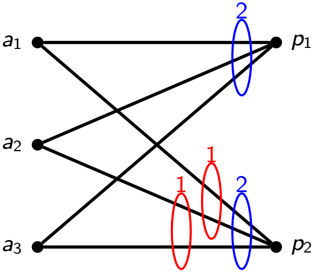


- Example: Countries , States , Districts , Cities
- Special case: Partition

Problem setup

Input:

- A set of applicants A
- A set of posts P
- ~~Applicants have preferences over a subset in~~ ~~P~~
- Posts have quotas
- Posts have laminar classes and quotas



Goal: Compute a maximum cardinality matching.

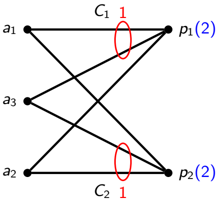
$\boxed{\text{Maximum matchings under laminar classifications}}$

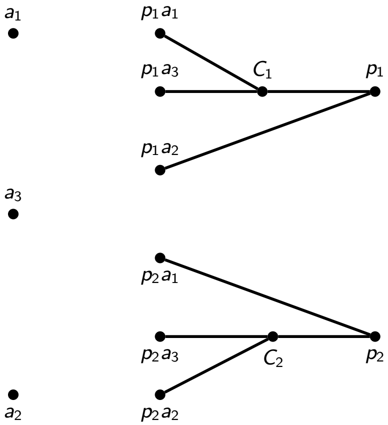
$\boxed{\text{Maximum matchings under laminar classifications}}$

$\Downarrow$

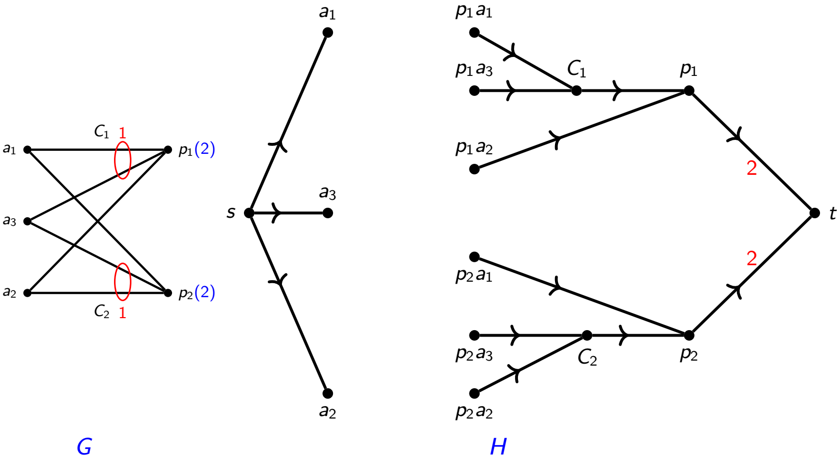
$\boxed{\text{Maximum flow in a flow network}}$

Classification tree - property of laminar classification



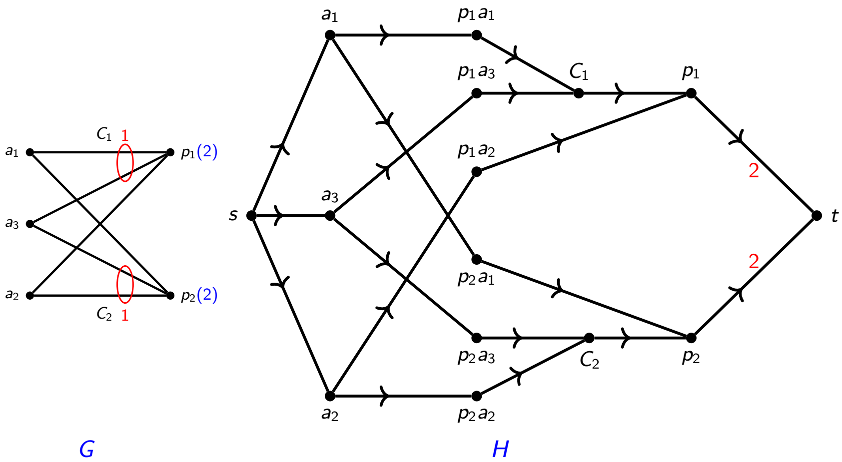


Feasible matchings to feasible flows



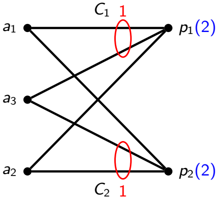
Add source s and sink t

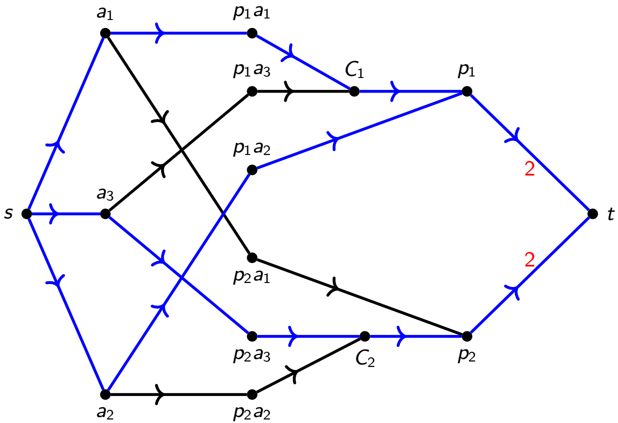
Feasible matchings to feasible flows



Add (a, pa) edges

Feasible matchings to feasible flows

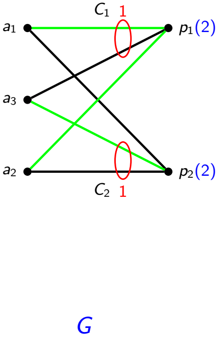


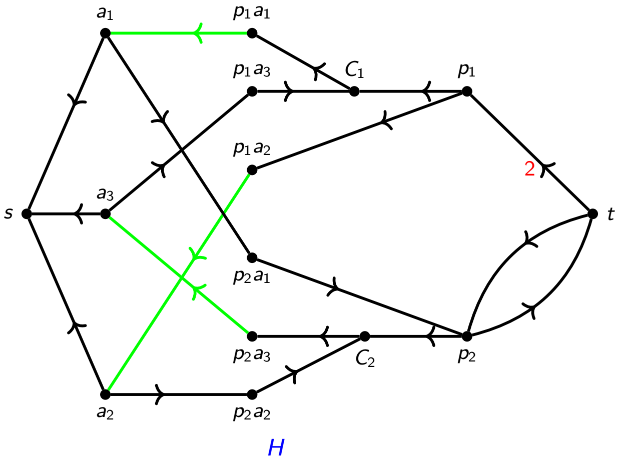


G H

$\boxed{\text{Compute a max-flow in } H}$

Feasible matchings to feasible flows





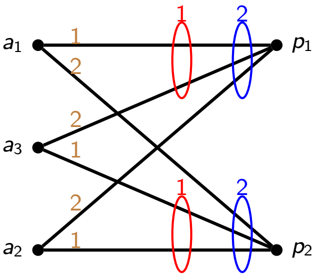
$\boxed{\text{Maximum matching in } G \iff \text{Maximum flow in } H}$

Back to our problem

Back to our problem

Input:

- A set of applicants A.
- A set of posts P.
- Applicants have preferences over a subset in P.
- Posts have quotas.
- Posts have laminar classes.

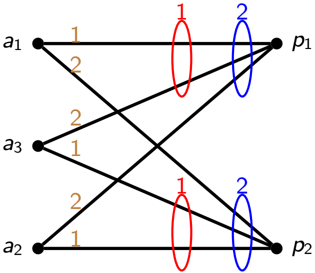


Goal: Match applicants to posts optimally.

Back to our problem

Input:

- A set of applicants A.
- A set of posts P.
- Applicants have preferences over a subset in P.
- Posts have quotas.
- Posts have laminar classes.



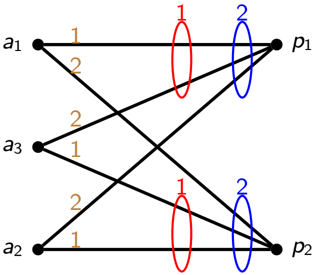
Goal: Match applicants to posts optimally.

Popularity: majority does not want to deviate.

Back to our problem

Input:

- A set of applicants A.
- A set of posts P.
- Applicants have preferences over a subset in P.
- Posts have quotas.
- Posts have laminar classes.



Goal: Match applicants to posts optimally.

Rank-maximality: max. number to rank-1, subject to this max. number to rank-2, ...

$\boxed{\text{Popularity in the one-to-one setting}}$

Popularity: definition

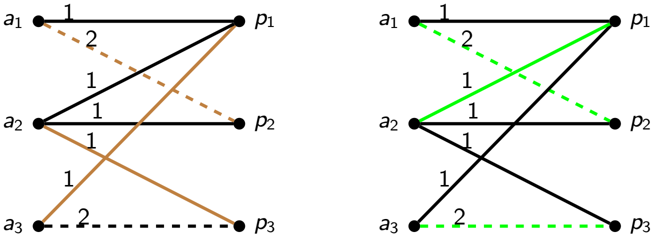
Gärdenfors (1975), Abraham et al.(2005)

Compare matchings M and M' using applicant votes.

Popularity: definition

Gärdenfors (1975), Abraham et al.(2005)

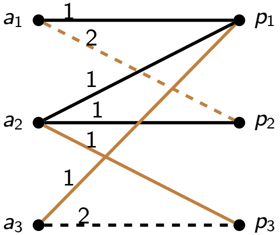
Compare matchings M and M' using applicant votes.

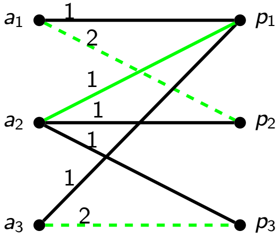


Popularity: definition

Gärdenfors (1975), Abraham et al.(2005)

Compare matchings M and M' using applicant votes.



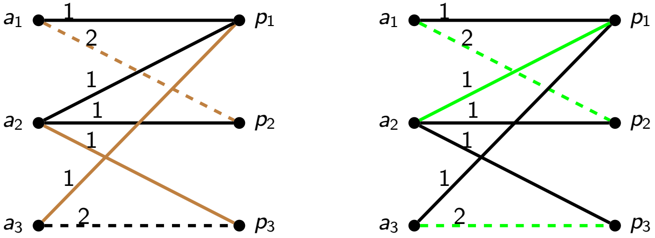


	M	M'
a_1	-	-
a_2	-	-
a_3	✓	-

Popularity: definition

Gärdenfors (1975), Abraham et al.(2005)

Compare matchings M and M' using applicant votes.



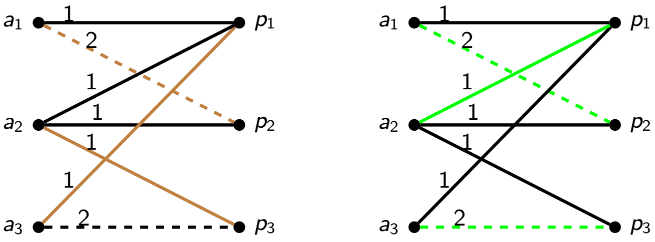
	M	M'
a_1	-	-
a_2	-	-
a_3	✓	-

M beats M' $\implies$ M' is not popular

Popularity: definition

Gärdenfors (1975), Abraham et al.(2005)

Compare matchings M and M' using applicant votes.



	M	M'
a_1	-	-
a_2	-	-
a_3	✓	-

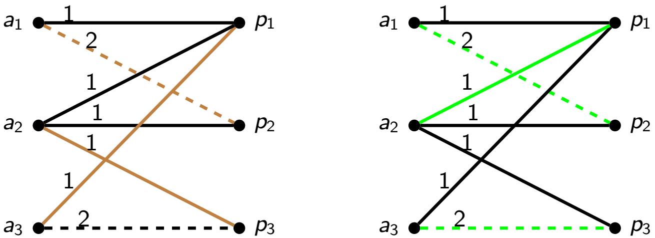
M beats M' $\implies$ M' is not popular

A matching M is popular if no matching beats it.

Popularity: definition

Gärdenfors (1975), Abraham et al.(2005)

Compare matchings M and M' using applicant votes.



	M	M'
a_1	-	-
a_2	-	-
a_3	✓	-

M beats M' $\implies$ M' is not popular

A matching M is popular if no matching beats it.

Popular matchings: characterization

Abraham et al. (2005)

$\boxed{\text{A matching } M \text{ is popular if no matching beats it.}}$

Popular matchings: characterization

Abraham et al. (2005)

$\boxed{\text{A matching } M \text{ is popular if no matching beats it.}}$

A matching M is popular if and only if:

- M is a maximum matching on the rank-1 edges.
- Every $a \in A$ is matched to either its $f(a)$ or $s(a)$

Popular matchings: characterization

Abraham et al. (2005)

A matching M is popular if no matching beats it.

A matching M is popular if and only if:

- M is a maximum matching on the rank-1 edges.
- Every $a \in A$ is matched to either its $f(a)$ or $s(a)$

$f(a)$ - set of all rank-1 posts of a.
$s(a)$ - next most preferred posts of a.

Popular matchings: characterization

Abraham et al. (2005)

$\boxed{\text{A matching } M \text{ is popular if no matching beats it.}}$

A matching M is popular if and only if:

- M is a maximum matching on the rank-1 edges.
- Every $a \in A$ is matched to either its $f(a)$ or $s(a)$

$f(a)$ - set of all rank-1 posts of a.
$s(a)$ - next most preferred posts of a.

- If such a matching does not exist, no popular matching exists.

Computing popular matchings Abraham et al. (2005); Manlove and Sng (2006)

Overall idea: Reduction to two maximum matching computations.

Computing popular matchings

Abraham et al. (2005); Manlove and Sng (2006)

Overall idea: Reduction to two maximum matching computations.

---

1. Construct G_1: every a adds $(a, f(a))$ edges.

   G_1 is graph on rank-1 edges.

Computing popular matchings

Abraham et al. (2005); Manlove and Sng (2006)

Overall idea: Reduction to two maximum matching computations.

---

1. Construct G_1: every a adds $(a, f(a))$ edges.

   G_1 is graph on rank-1 edges.

2. Compute a maximum matching M_1 in G_1.
3. Delete "unnecessary" rank-1 edges.

Computing popular matchings

Abraham et al. (2005); Manlove and Sng (2006)

Overall idea: Reduction to two maximum matching computations.

---

1. Construct G_1: every a adds $(a, f(a))$ edges.

   G_1 is graph on rank-1 edges.

2. Compute a maximum matching M_1 in G_1.
3. Delete "unnecessary" rank-1 edges.
4. Some applicants add $(a, s(a))$ edges.

Computing popular matchings

Abraham et al. (2005); Manlove and Sng (2006)

Overall idea: Reduction to two maximum matching computations.

---

1. Construct G_1: every a adds $(a, f(a))$ edges.

   G_1 is graph on rank-1 edges.

2. Compute a maximum matching M_1 in G_1.
3. Delete "unnecessary" rank-1 edges.
4. Some applicants add $(a, s(a))$ edges.
5. Augment M_1 to compute a maximum matching M.

Computing popular matchings

Abraham et al. (2005); Manlove and Sng (2006)

Overall idea: Reduction to two maximum matching computations.

---

1. Construct G_1: every a adds $(a, f(a))$ edges.

   G_1 is graph on rank-1 edges.
2. Compute a maximum matching M_1 in G_1.
3. Delete "unnecessary" rank-1 edges.
4. Some applicants add $(a, s(a))$ edges.
5. Augment M_1 to compute a maximum matching M.
6. If M matches all applicants, declare popular,
   else no popular matching.

Computing popular matchings

Abraham et al. (2005); Manlove and Sng (2006)

Overall idea: Reduction to two maximum matching computations.

1. Construct G_1: every a adds $(a, f(a))$ edges.
   G_1 is graph on rank-1 edges.
2. Compute a maximum matching M_1 in G_1.
3. Delete "unnecessary" rank-1 edges.
4. Some applicants add $(a, s(a))$ edges.
5. Augment M_1 to compute a maximum matching M.
6. If M matches all applicants, declare popular,
   else no popular matching.

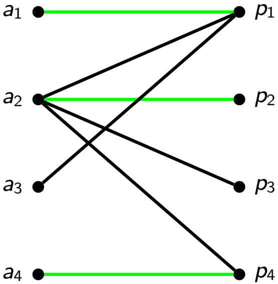
Steps 3 & 4: Dulmage Mendelsohn Decomposition

Dulmage Mendelsohn decomposition DM (1958)

Partition of vertices into three sets w.r.t. a maximum matching.

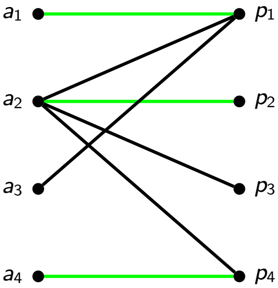
Dulmage Mendelsohn decomposition DM (1958)

Partition of vertices into three sets w.r.t. a maximum matching.



Dulmage Mendelsohn decomposition DM (1958)

Partition of vertices into three sets w.r.t. a maximum matching.

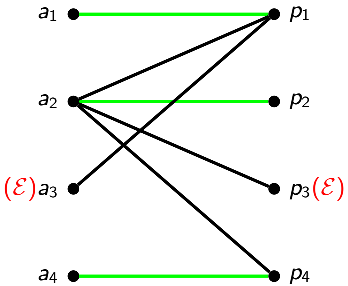


M is a maximum matching

Dulmage Mendelsohn decomposition

DM (1958)

Partition of vertices into three sets w.r.t. a maximum matching.

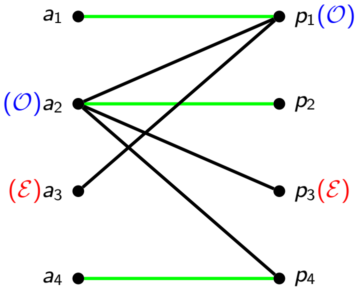


M is a maximum matching

- $\mathcal{E}$: reachable from unmatched vertex via even length alt. path.

Dulmage Mendelsohn decomposition DM (1958)

Partition of vertices into three sets w.r.t. a maximum matching.

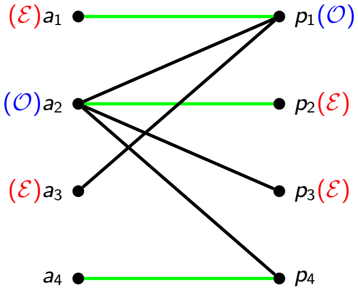


M is a maximum matching

- $\mathcal{E}$: reachable from unmatched vertex via even length alt. path.
- $\mathcal{O}$: reachable from unmatched vertex via odd length alt. path.

Dulmage Mendelsohn decomposition DM (1958)

Partition of vertices into three sets w.r.t. a maximum matching.



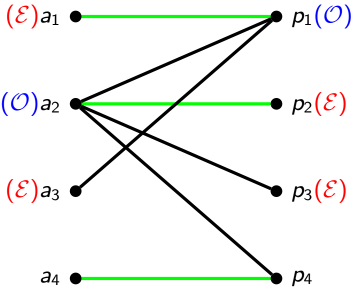
M is a maximum matching

- $\mathcal{E}$: reachable from unmatched vertex via even length alt. path.
- $\mathcal{O}$: reachable from unmatched vertex via odd length alt. path.

Dulmage Mendelsohn decomposition

DM (1958)

Partition of vertices into three sets w.r.t. a maximum matching.

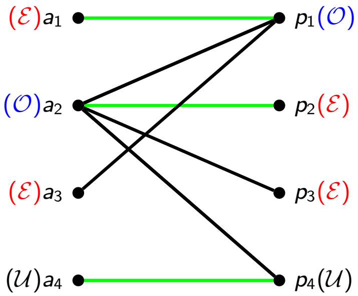


M is a maximum matching

- $\mathcal{E}$: reachable from unmatched vertex via even length alt. path.
- $\mathcal{O}$: reachable from unmatched vertex via odd length alt. path.

Dulmage Mendelsohn decomposition DM (1958)

Partition of vertices into three sets w.r.t. a maximum matching.

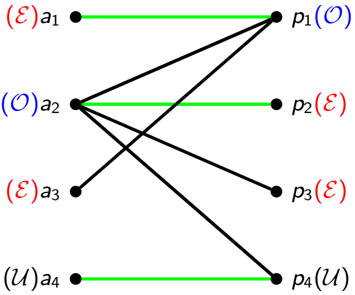


M is a maximum matching

- $\mathcal{E}$: reachable from unmatched vertex via even length alt. path.
- $\mathcal{O}$: reachable from unmatched vertex via odd length alt. path.
- $\mathcal{U}$: **unreachable** from unmatched vertex via length alt. path.

Dulmage Mendelsohn decomposition DM (1958)

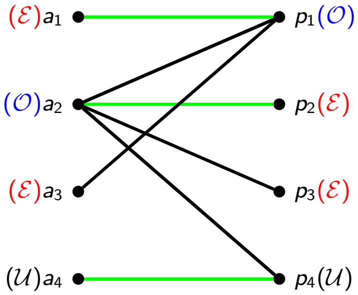
Partition of vertices into three sets w.r.t. a maximum matching.



Dulmage Mendelsohn decomposition

DM (1958)

Partition of vertices into three sets w.r.t. a maximum matching.

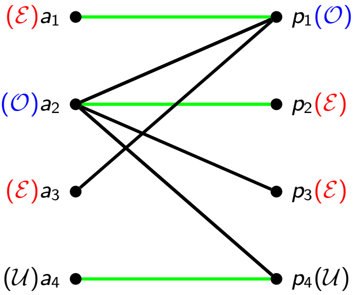


$\boxed{\mathcal{E}, \mathcal{O} \text{ and } \mathcal{U} \text{ invariant of } M}$

Dulmage Mendelsohn decomposition

DM (1958)

Partition of vertices into three sets w.r.t. a maximum matching.



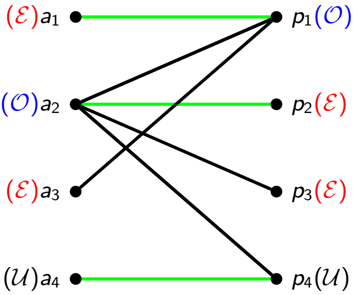
$\boxed{\mathcal{E}, \mathcal{O} \text{ and } \mathcal{U} \text{ invariant of } M}$

For any maximum matching:

Dulmage Mendelsohn decomposition

DM (1958)

Partition of vertices into three sets w.r.t. a maximum matching.



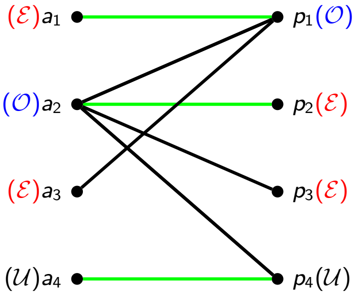
$\boxed{\mathcal{E}, \mathcal{O} \text{ and } \mathcal{U} \text{ invariant of } M}$

For any maximum matching:

- Every $\mathcal{O}$ and $\mathcal{U}$ vertex is matched.

Dulmage Mendelsohn decomposition DM (1958)

Partition of vertices into three sets w.r.t. a maximum matching.



$\boxed{\mathcal{E},\ \mathcal{O} \text{ and } \mathcal{U} \text{ invariant of } M}$

For any maximum matching:

- Every $\mathcal{O}$ and $\mathcal{U}$ vertex is matched.
- No $\mathcal{OO}$,$\mathcal{OU}$ edges are matched.

Computing popular matchings

Abraham et al. (2005); Manlove and Sng (2006)

Overall idea: Reduction to two maximum matching computations.

Computing popular matchings

Abraham et al. (2005); Manlove and Sng (2006)

Overall idea: Reduction to two maximum matching computations.

---

1. Construct G_1: every a adds $(a, f(a))$ edges.

   G_1 is graph on rank-1 edges.

2. Compute a maximum matching M_1 in G_1.
3. Delete "unnecessary" rank-1 edges. $\mathcal{OO}$,$\mathcal{OU}$ edges.
4. Even applicants add $(a, s(a))$ edges.
5. Augment M_1 to compute a maximum matching M.
6. If M matches all applicants, declare popular,
   else no popular matching.

Computing popular matchings

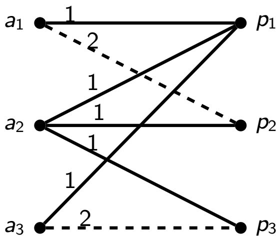
Abraham et al. (2005); Manlove and Sng (2006)

Overall idea: Reduction to two maximum matching computations.

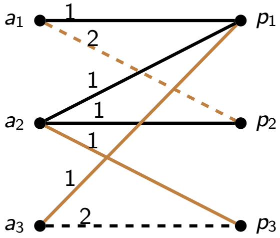
1. Construct G_1: every a adds $(a, f(a))$ edges.
   G_1 is graph on rank-1 edges.
2. Compute a maximum matching M_1 in G_1.
3. Delete "unnecessary" rank-1 edges. $\mathcal{OO}$,$\mathcal{OU}$ edges.
4. Even applicants add $(a, s(a))$ edges.
5. Augment M_1 to compute a maximum matching M.
6. If M matches all applicants, declare popular,
   else no popular matching.

$\boxed{\text{Step 4: } s(a) \text{ – most preferred even post in } G_1.}$

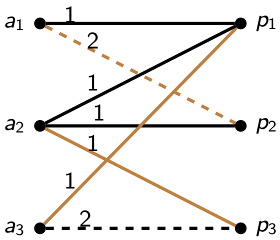
Is deletion necessary?

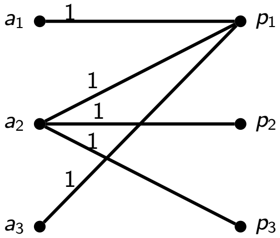


Is deletion necessary?



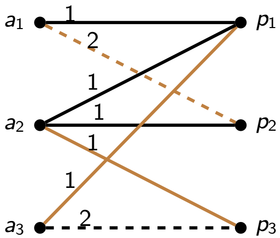
Is deletion necessary?

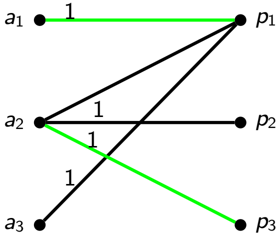




Graph G_1

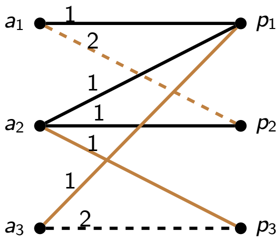
Is deletion necessary?

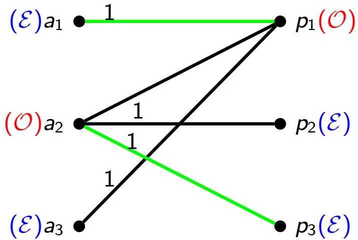




Max. matching M_1 in G_1

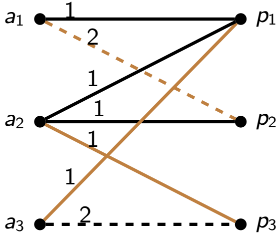
Is deletion necessary?

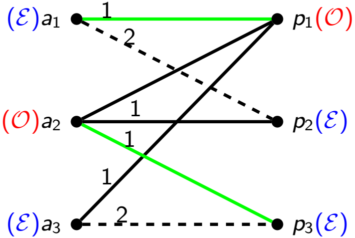




Label vertices

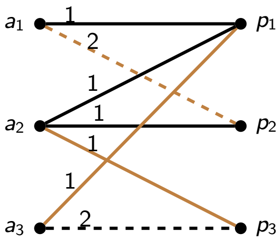
Is deletion necessary?

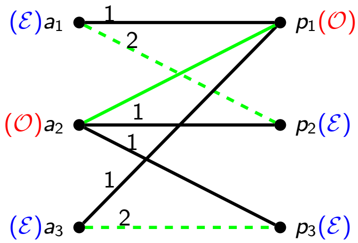




Add $(a, s(a))$ edges

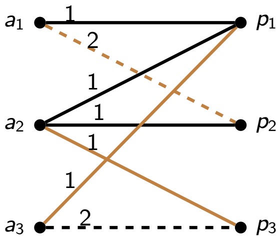
Is deletion necessary?

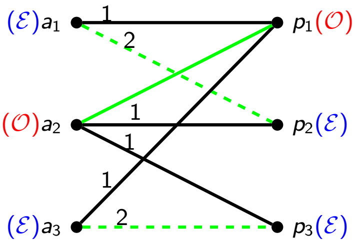




Augment M_1 to get M'

Is deletion necessary?

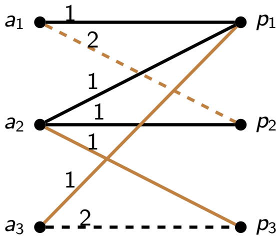


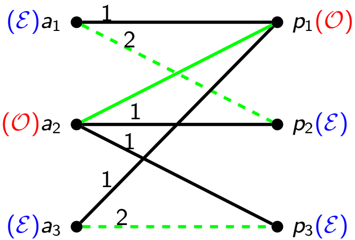


	M	M'
a_1	-	-
a_2	-	-
a_3	✓	-

M' is not popular

Is deletion necessary?





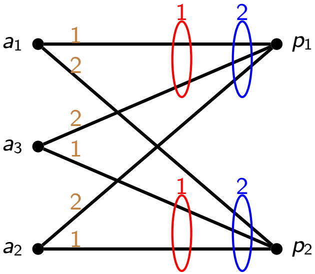
Deletion of $\mathcal{OO}$, $\mathcal{OU}$ edges is crucial!

Back to our problem

Back to our problem

Input:

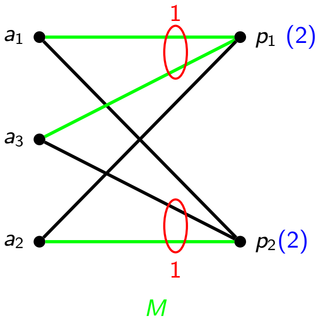
- A set of applicants A.
- A set of posts P.
- Applicants have preferences over a subset in P.
- Posts have quotas.
- Posts have laminar classes.



Goal: Compute a popular matching of applicants to posts (if one exists).

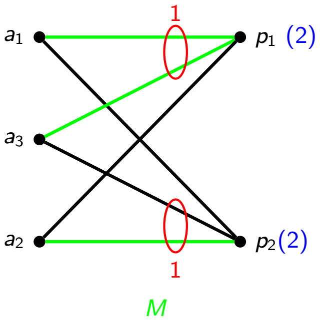
$\boxed{\text{Laminar classified popular matchings}}$

Classified matchings: challenges



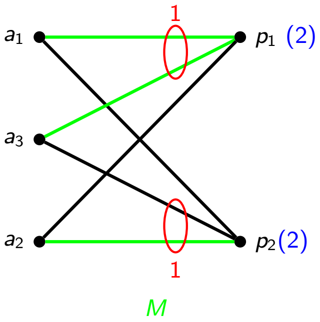
- Deal with capacitated matchings.

Classified matchings: challenges



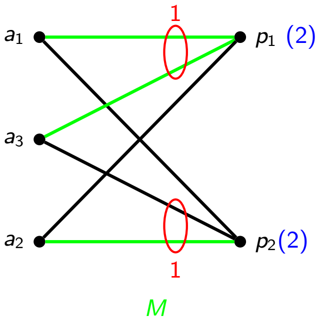
- Deal with capacitated matchings.
  - Manlove and Sng use *cloning.*
  - Paluch defined *good paths.*

Classified matchings: challenges



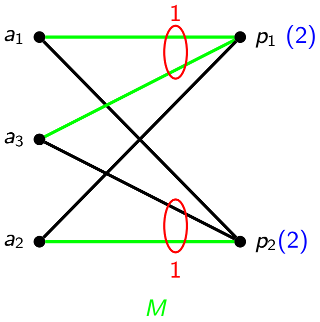
- Deal with capacitated matchings.
  - Manlove and Sng use *cloning.*
  - Paluch defined *good paths.*
- Both techniques do not work for classifications.

Classified matchings: challenges



- Deal with capacitated matchings.
  - Manlove and Sng use *cloning.*
  - Paluch defined *good paths.*
- Both techniques do not work for classifications.
  - Maximum matching M not feasible.

Classified matchings: challenges



- Deal with capacitated matchings.
  - Manlove and Sng use *cloning.*
  - Paluch defined *good paths.*
- Both techniques do not work for classifications.
  - Maximum matching M not feasible.

Use max-flow and min-cut properties!

Properties of max-flow

Let H be any flow network and f be a max-flow in H.

$$\begin{aligned} S &= \{v \mid \text{ is reachable from } s \text{ in } H_f\} \\ T &= \{v \mid v \text{ can reach } t \text{ in } H_f\} \\ U &= \{v \mid v \notin T \cup S\} \end{aligned}$$

Properties of max-flow

Let H be any flow network and f be a max-flow in H.

$$
\begin{aligned}
S &= \{v \mid \text{ is reachable from } s \text{ in } H_f\} \\
T &= \{v \mid v \text{ can reach } t \text{ in } H_f\} \\
U &= \{v \mid v \notin T \cup S\}
\end{aligned}
$$

$$\boxed{(S, T \cup U) \text{ is a min-s-t-cut in } H}$$

Properties of max-flow

Let H be any flow network and f be a max-flow in H.

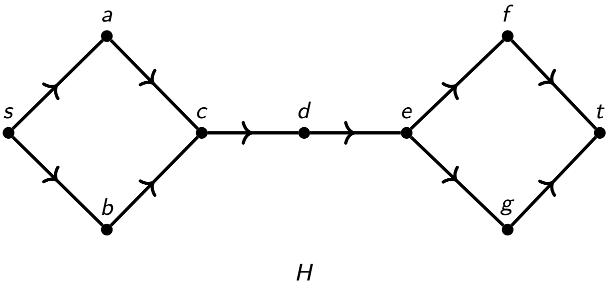
$$\begin{aligned} S &= \{v \mid \text{ is reachable from } s \text{ in } H_f\} \\ T &= \{v \mid v \text{ can reach } t \text{ in } H_f\} \\ U &= \{v \mid v \notin T \cup S\} \end{aligned}$$

$$\boxed{(S, T \cup U) \text{ is a min-s-t-cut in } H}$$

Known Facts:

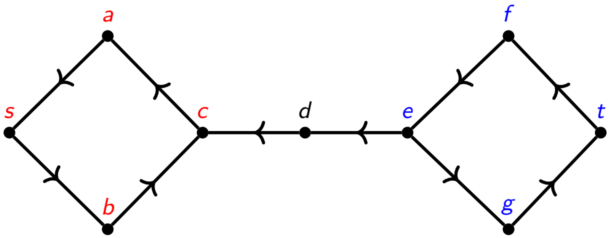
- Forward edges $(S, T \cup U)$: saturated in **every** max-flow.
- Reverse edges $(T \cup U, S)$: *zero* flow in **every** max-flow.

Properties of max-flow



$$\begin{aligned} S &= \{v \mid \text{ is reachable from } s \text{ in } H_f\} \\ T &= \{v \mid v \text{ can reach } t \text{ in } H_f\} \\ U &= \{v \mid v \notin T \cup S\} \end{aligned}$$

Properties of max-flow



H_f

$$
\begin{aligned}
S &= \{v \mid \text{ is reachable from } s \text{ in } H_f\} \\
T &= \{v \mid v \text{ can reach } t \text{ in } H_f\} \\
U &= \{v \mid v \notin T \cup S\}
\end{aligned}
$$

Properties of max-flow

The sets S, T and U are **invariant** of the max-flow f.

Properties of max-flow

$\boxed{\text{The sets } S,\ T \text{ and } U \text{ are } \textbf{invariant} \text{ of the max-flow } f.}$

(i) $x \in S_f \iff x \in S_{f'}$

Properties of max-flow

$\boxed{\text{The sets } S\text{, } T \text{ and } U \text{ are } \textbf{invariant} \text{ of the max-flow } f.}$

(i) $x \in S_f \iff x \in S_{f'}$

Proof: $x \in S_f$ and $x \in T_{f'} \cup U_{f'}$

Properties of max-flow

$\boxed{\text{The sets } S,\ T \text{ and } U \text{ are \textbf{invariant} of the max-flow } f.}$

(i) $x \in S_f \iff x \in S_{f'}$

Proof: $x \in S_f$ and $x \in T_{f'} \cup U_{f'}$
x be the nearest such node from s in H_f.

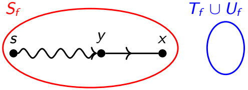
Properties of max-flow

$\boxed{\text{The sets } S,\ T \text{ and } U \text{ are } \textbf{invariant} \text{ of the max-flow } f.}$

(i) $x \in S_f \iff x \in S_{f'}$

Proof: $x \in S_f$ and $x \in T_{f'} \cup U_{f'}$

x be the nearest such node from s in H_f.



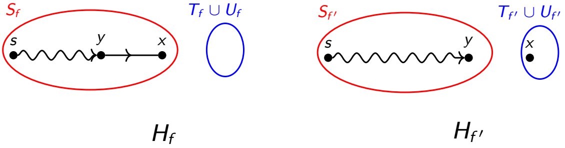
H_f

Properties of max-flow

The sets S, T and U are **invariant** of the max-flow f.

(i) $x \in S_f \iff x \in S_{f'}$

Proof: $x \in S_f$ and $x \in T_{f'} \cup U_{f'}$
x be the nearest such node from s in H_f.



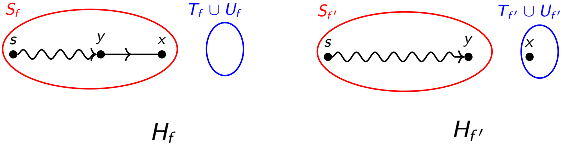
- $(y, x) \in H$
- $(x, y) \in H$

Properties of max-flow

The sets S, T and U are **invariant** of the max-flow f.

(i) $x \in S_f \iff x \in S_{f'}$

Proof: $x \in S_f$ and $x \in T_{f'} \cup U_{f'}$
x be the nearest such node from s in H_f.



- $(y, x) \in H \implies f(y, x) = f'(y, x) = c(y, x)$. $\Rightarrow\Leftarrow$
- $(x, y) \in H$

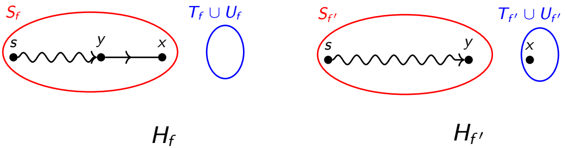
Properties of max-flow

The sets S, T and U are **invariant** of the max-flow f.

(i) $x \in S_f \iff x \in S_{f'}$

Proof: $x \in S_f$ and $x \in T_{f'} \cup U_{f'}$

x be the nearest such node from s in H_f.



- $(y, x) \in H \implies f(y, x) = f'(y, x) = c(y, x).$ $\Rightarrow\Leftarrow$
- $(x, y) \in H \implies f(x, y) = f'(x, y) = 0.$ $\Rightarrow\Leftarrow$

Classified popular matchings: characterization

A matching M is popular if no matching beats it.

Classified popular matchings: characterization

A matching M is popular if no matching beats it.

A matching M is popular if and only if:

- M is a maximum feasible matching on the rank-1 edges.
- Every $a \in A$ is matched to either its $f(a)$ or $s(a)$.

Classified popular matchings: characterization

A matching M is popular if no matching beats it.

A matching M is popular if and only if:

- M is a maximum feasible matching on the rank-1 edges.
- Every $a \in A$ is matched to either its $f(a)$ or $s(a)$.

$f(a)$ - set of all rank-1 posts of a.
$s(a)$ - defined using flow network on rank-1 edges.

Classified popular matchings: characterization

A matching M is popular if no matching beats it.

A matching M is popular if and only if:

- M is a maximum feasible matching on the rank-1 edges.
- Every $a \in A$ is matched to either its $f(a)$ or $s(a)$.

$f(a)$ - set of all rank-1 posts of a.
$s(a)$ - defined using flow network on rank-1 edges.

- If such a matching does not exist, no popular matching exists.

Computing classified popular matchings

Overall idea: Reduction to two maximum flow computations.

Computing classified popular matchings

Overall idea: Reduction to two maximum flow computations.

---

1. Construct H_1: every a adds $(a, f(a))$ edges.

Computing classified popular matchings

Overall idea: Reduction to two maximum flow computations.

---

1. Construct H_1: every a adds $(a, f(a))$ edges.
2. Compute a maximum flow f_1 in H_1.
3. Delete $(T \cup U, S)$ edges in $H_1(f_1)$.

Computing classified popular matchings

Overall idea: Reduction to two maximum flow computations.

---

1. Construct H_1: every a adds $(a, f(a))$ edges.
2. Compute a maximum flow f_1 in H_1.
3. Delete $(T \cup U, S)$ edges in $H_1(f_1)$.

   Ensures that augmentation preserves max. card. on rank-1.

Computing classified popular matchings

Overall idea: Reduction to two maximum flow computations.

---

1. Construct H_1: every a adds $(a, f(a))$ edges.
2. Compute a maximum flow f_1 in H_1.
3. Delete $(T \cup U, S)$ edges in $H_1(f_1)$.

   Ensures that augmentation preserves max. card. on rank-1.
4. For every $a \in S$, add $(a, s(a))$ edge.

   $s(a)$ is the most preferred post $p \in T$ in $H_1(f_1)$.

Computing classified popular matchings

Overall idea: Reduction to two maximum flow computations.

---

1. Construct H_1: every a adds $(a, f(a))$ edges.
2. Compute a maximum flow f_1 in H_1.
3. Delete $(T \cup U, S)$ edges in $H_1(f_1)$.

   Ensures that augmentation preserves max. card. on rank-1.
4. For every $a \in S$, add $(a, s(a))$ edge.

   $s(a)$ is the most preferred post $p \in T$ in $H_1(f_1)$.
5. Augment f_1 to obtain f_2. Let M be matching corr. to f_2.

Computing classified popular matchings

Overall idea: Reduction to two maximum flow computations.

---

1. Construct H_1: every a adds $(a, f(a))$ edges.
2. Compute a maximum flow f_1 in H_1.
3. Delete $(T \cup U, S)$ edges in $H_1(f_1)$.

   Ensures that augmentation preserves max. card. on rank-1.
4. For every $a \in S$, add $(a, s(a))$ edge.

   $s(a)$ is the most preferred post $p \in T$ in $H_1(f_1)$.
5. Augment f_1 to obtain f_2. Let M be matching corr. to f_2.
6. If M matches all applicants, declare popular, else no popular matching.

Computing classified popular matchings

Overall idea: Reduction to two maximum flow computations.

1. Construct H_1: every a adds $(a, f(a))$ edges.
2. Compute a maximum flow f_1 in H_1.
3. Delete $(T \cup U, S)$ edges in $H_1(f_1)$.
   Ensures that augmentation preserves max. card. on rank-1.
4. For every $a \in S$, add $(a, s(a))$ edge.
   $s(a)$ is the most preferred post $p \in T$ in $H_1(f_1)$.
5. Augment f_1 to obtain f_2. Let M be matching corr. to f_2.
6. If M matches all applicants, declare popular,
   else no popular matching.

- An $O(|A| \cdot |E|)$ time algorithm.

Classified popular matchings – correctness

A matching M is popular if and only if:

- M is a maximum feasible matching on the rank-1 edges.
- Every $a \in A$ is matched to either its $f(a)$ or $s(a)$.

- If such a matching does not exist, no popular matching exists.

Classified popular matchings – correctness

A matching M is popular if and only if:

- M is a maximum feasible matching on the rank-1 edges.
- Every $a \in A$ is matched to either its $f(a)$ or $s(a)$.

- If such a matching does not exist, no popular matching exists.

Proof technique:

1. Two promotions at the cost of one demotion.
2. Cut edges were deleted $\implies$ augmentation preserves rank-1 edges.

To summarize ..

- A flow based framework for popular matchings with classifications.

To summarize ..

- A flow based framework for popular matchings with classifications.
- Same framework applies for rank-maximal matchings.
  Classifications are allowed on both sides.

To summarize ..

- A flow based framework for popular matchings with classifications.
- Same framework applies for rank-maximal matchings.
  Classifications are allowed on both sides.
- Key step: Use max-flow, min-cut properties to identify "unnecessary" edges.

To summarize ..

- A flow based framework for popular matchings with classifications.
- Same framework applies for rank-maximal matchings.
  Classifications are allowed on both sides.
- Key step: Use max-flow, min-cut properties to identify "unnecessary" edges.
- Laminarity is needed, else problem is NP-Hard.

To summarize ..

- A flow based framework for popular matchings with classifications.
- Same framework applies for rank-maximal matchings.
  Classifications are allowed on both sides.
- Key step: Use max-flow, min-cut properties to identify “unnecessary” edges.
- Laminarity is needed, else problem is NP-Hard.

Extensions:

- Popular matchings in the many to many settings.
- Lower quotas for classes.

To summarize ..

- A flow based framework for popular matchings with classifications.
- Same framework applies for rank-maximal matchings.
  Classifications are allowed on both sides.
- Key step: Use max-flow, min-cut properties to identify "unnecessary" edges.
- Laminarity is needed, else problem is NP-Hard.

Extensions:

- Popular matchings in the many to many settings.
- Lower quotas for classes.

Thank you!