An Efficient Algorithm for the "Optimal" Stable Marriage

Based on the work by Robert W. Irving, Paul Leather, and Dan Gusfield

Sankar Vinayak Vaibhav

Department of Computer Science and Engineering
Indian Institute of Technology Madras

23 April 2025

Stable Marriage Problem

Consider a complete bipartite graph with two sets A and B, each containing n elements. Every element in A ranks all elements in B in strict order of preference, and vice versa. A matching is said to be **stable** if there is no pair (a, b) such that:

- $a \in A$ prefers $b \in B$ over its current match, and
- b prefers a over its current match.

Such a pair (a, b) would be called a *blocking pair*.

Gale and Shapley

Gale and Shapley Matching Algorithm:

1. **Initialize preferences:** Each element in sets A and B (two sides of a bipartite graph) has a strict preference list over elements of the opposite set.
2. **Proposals:** Each free element in A proposes to the most preferred element in B that has not yet rejected it.
3. **Acceptance:** Each element in B tentatively accepts the most preferred proposal it has received so far and rejects the rest.
4. **Repeat:** Continue proposals and acceptances until every element in A is matched or no further proposals are possible.

This results in a stable matching between A and B.

Is Gale and Shapley biased?

Lets define preference ordering for both the sides

A Preferences

a_1: b_1, b_3, b_2, b_4
a_2: b_2, b_4, b_1, b_3
a_3: b_3, b_1, b_4, b_2
a_4: b_4, b_2, b_3, b_1

B Preferences

b_1: a_4, a_3, a_2, a_1
b_2: a_3, a_4, a_1, a_2
b_3: a_2, a_1, a_4, a_3
b_4: a_1, a_2, a_3, a_4

Is Gale and Shapley biased?

Lets define preference ordering for both the sides

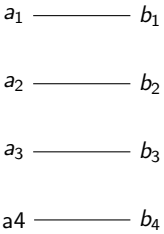
A Preferences

a_1: b_1, b_3, b_2, b_4
a_2: b_2, b_4, b_1, b_3
a_3: b_3, b_1, b_4, b_2
a_4: b_4, b_2, b_3, b_1

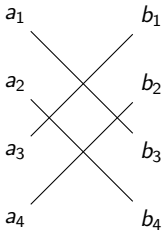
B Preferences

b_1: a_4, a_3, a_2, a_1
b_2: a_3, a_4, a_1, a_2
b_3: a_2, a_1, a_4, a_3
b_4: a_1, a_2, a_3, a_4

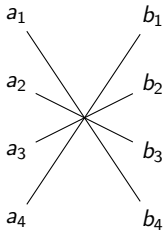
For this preferences lets see 3 Stable matchings



A Optimal



M



B Optimal

How to Measure the Quality of a Matching?

We measure quality based on rankings in preference lists:

$$ar(i,k) = j \quad : \text{if } k \in B \text{ is the } j^{\text{th}} \text{ choice of } i \in A$$
$$br(i,k) = j \quad : \text{if } k \in A \text{ is the } j^{\text{th}} \text{ choice of } i \in B$$

How to Measure the Quality of a Matching?

We measure quality based on rankings in preference lists:

$$ar(i,k) = j \quad : \text{if } k \in B \text{ is the } j^{\text{th}} \text{ choice of } i \in A$$
$$br(i,k) = j \quad : \text{if } k \in A \text{ is the } j^{\text{th}} \text{ choice of } i \in B$$

Given a stable matching $S = \{(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)\}$, we define the **cost** of the matching as:

$$c(S) = \sum_{i=1}^{n} ar(a_i, b_i) + \sum_{i=1}^{n} br(b_i, a_i)$$

How to Measure the Quality of a Matching?

We measure quality based on rankings in preference lists:

$$ar(i,k) = j \quad : \text{if } k \in B \text{ is the } j^{\text{th}} \text{ choice of } i \in A$$
$$br(i,k) = j \quad : \text{if } k \in A \text{ is the } j^{\text{th}} \text{ choice of } i \in B$$

Given a stable matching $S = \{(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)\}$, we define the **cost** of the matching as:

$$c(S) = \sum_{i=1}^{n} ar(a_i, b_i) + \sum_{i=1}^{n} br(b_i, a_i)$$

A stable matching is called **egalitarian-optimal** if it minimizes the total cost $c(S)$.

Observation: The minimum possible value of $c(S)$ is $2n$, which occurs when everyone gets their first choice.

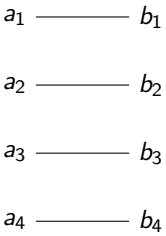
Satisfaction computation

A Preferences

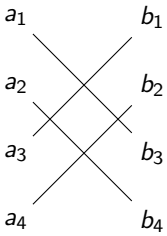
a_1: b_1, b_3, b_2, b_4
a_2: b_2, b_4, b_1, b_3
a_3: b_3, b_1, b_4, b_2
a_4: b_4, b_2, b_3, b_1

B Preferences

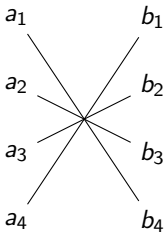
b_1: a_4, a_3, a_2, a_1
b_2: a_3, a_4, a_1, a_2
b_3: a_2, a_1, a_4, a_3
b_4: a_1, a_2, a_3, a_4



C(A Optimal)=20



C(M)=16



C(B Optimal)=20

Example of shortlists.

A Preferences

a_1: b_3, b_1, b_5, b_4, b_2
a_2: b_1, b_3, b_4, b_5, b_2
a_3: b_4, b_3, b_5, b_1, b_2
a_4: b_5, b_3, b_1, b_2, b_4
a_5: b_4, b_1, b_2, b_3, b_5

B Preferences

b_1: a_4, a_3, a_1, a_2, a_5
b_2: a_3, a_5, a_4, a_1, a_2
b_3: a_5, a_3, a_2, a_1, a_4
b_4: a_4, a_2, a_3, a_1, a_5
b_5: a_1, a_5, a_4, a_3, a_2

Example of shortlists.

A Preferences

a_1: b_3, b_1, b_5, b_4, b_2
a_2: b_1, b_3, b_4, b_5, b_2
a_3: b_4, b_3, b_5, b_1, b_2
a_4: b_5, b_3, b_1, b_2, b_4
a_5: b_4, b_1, b_2, b_3, b_5

B Preferences

b_1: a_4, a_3, a_1, a_2, a_5
b_2: a_3, a_5, a_4, a_1, a_2
b_3: a_5, a_3, a_2, a_1, a_4
b_4: a_4, a_2, a_3, a_1, a_5
b_5: a_1, a_5, a_4, a_3, a_2

Example of shortlists.

A Preferences

a_1: b_3, b_1, b_5, b_4, b_2
a_2: b_1, b_3, b_4, b_5, b_2
a_3: b_4, b_3, b_5, b_1, b_2
a_4: b_5, b_3, b_1, b_2, b_4
a_5: b_4, b_1, b_2, b_3, b_5

B Preferences

b_1: a_4, a_3, a_1, a_2, a_5
b_2: a_3, a_5, a_4, a_1, a_2
b_3: a_5, a_3, a_2, a_1, a_4
b_4: a_4, a_2, a_3, a_1, a_5
b_5: a_1, a_5, a_4, a_3, a_2

Example of shortlists.

A Preferences

a_1: b_3, b_1, b_5, b_4, b_2
a_2: b_1, b_3, b_4, b_5, b_2
a_3: b_4, b_3, b_5, b_1, b_2
a_4: b_5, b_3, b_1, b_2, b_4
a_5: b_4, b_1, b_2, b_3, b_5

B Preferences

b_1: a_4, a_3, a_1, a_2, a_5
b_2: a_3, a_5, a_4, a_1, a_2
b_3: a_5, a_3, a_2, a_1, a_4
b_4: a_4, a_2, a_3, a_1, a_5
b_5: a_1, a_5, a_4, a_3, a_2

Example of shortlists.

A Preferences

a_1: b_3, b_1, b_5, ~~b_4, b_2~~
a_2: b_1, b_3, b_4, ~~b_5, b_2~~
a_3: b_4, b_3, ~~b_5~~, b_1, b_2
a_4: b_5, ~~b_3~~, b_1, ~~b_2~~, b_4
a_5: ~~b_4, b_1~~, b_2, b_3, b_5

B Preferences

b_1: a_4, a_3, a_1, a_2, ~~a_5~~
b_2: a_3, a_5, ~~a_4, a_1, a_2~~
b_3: a_5, a_3, a_2, a_1, ~~a_4~~
b_4: a_4, a_2, a_3, ~~a_1, a_5~~
b_5: a_1, a_5, a_4, ~~a_3, a_2~~

Example of shortlists.

A Preferences

a_1: b_3, b_1, b_5, ~~b_4, b_2~~
a_2: b_1, b_3, b_4, ~~b_5, b_2~~
a_3: b_4, b_3, ~~b_5~~, b_1, b_2
a_4: b_5, ~~b_3~~, b_1, ~~b_2~~, b_4
a_5: ~~b_4, b_1~~, b_2, b_3, b_5

B Preferences

b_1: a_4, a_3, a_1, a_2, ~~a_5~~
b_2: a_3, a_5, ~~a_4, a_1, a_2~~
b_3: a_5, a_3, a_2, a_1, ~~a_4~~
b_4: a_4, a_2, a_3, ~~a_1, a_5~~
b_5: a_1, a_5, a_4, ~~a_3, a_2~~

Cost of matching: $C(S) = 1 + 1 + 1 + 1 + 3 + 4 + 2 + 4 + 3 + 3 = 23$

Shortlist

Definition (Shortlist)

Given a stable marriage instance and a run of the Gale–Shapley algorithm (with one side proposing), the **shortlist** of an individual is the reduced preference list formed as follows:

- For a proposer (A), the shortlist consists of all individuals (B) to whom (A) proposed and were not rejected, or for whom (A) remains on (B)'s preference list.
- For a receiver (B), the shortlist consists of all proposers (A) who are preferred over any currently matched partner, as well as the currently matched (A).

Property 1: Stability and Shortlists

Property 1: If b does not appear on a's shortlist, then there is no stable matching in which a and b are partners.

Property 1: Stability and Shortlists

Property 1: If b does not appear on a's shortlist, then there is no stable matching in which a and b are partners.

- If a person is not included in the shortlist of another person, it is impossible for them to be paired in a stable matching.
- This ensures that only those who are "considered" can form a potential stable pair.

Property 2: Symmetry of Shortlists

Property 2: b appears on a's shortlist if and only if a appears on b's, and b is first on a's shortlist if and only if a is last on b's.

Property 2: Symmetry of Shortlists

Property 2: b appears on a's shortlist if and only if a appears on b's, and b is first on a's shortlist if and only if a is last on b's.

- The appearance of a partner in each other's shortlist is symmetric.
- This implies a certain balance in how candidates are evaluated by both sides, with preferences mirroring each other.

Property 3: A-Optimal Solution

Property 3: If each element in *a* is matched with the first choice on its shortlist, the resulting matching is stable. This is referred to as the **A-optimal** solution.

Property 3: A-Optimal Solution

Property 3: If each element in a is matched with the first choice on its shortlist, the resulting matching is stable. This is referred to as the **A-optimal** solution.

- This matching is optimal for A, since no element in A can be matched with a more preferred partner.
- Moreover, no element in B can be matched with a less preferred partner in any other stable matching.

Property 3: A-Optimal Solution

Property 3: If each element in a is matched with the first choice on its shortlist, the resulting matching is stable. This is referred to as the **A-optimal** solution.

- This matching is optimal for A, since no element in A can be matched with a more preferred partner.
- Moreover, no element in B can be matched with a less preferred partner in any other stable matching.

Key Point: This solution guarantees maximum satisfaction for the side A, while preserving stability.

Property 4: B-Optimal Solution

Property 4: If the roles of A and B are interchanged, and each element in B is matched with the first choice on its (B-oriented) shortlist, then the resulting matching is stable. This is called the **B-optimal** solution.

Property 4: B-Optimal Solution

Property 4: If the roles of A and B are interchanged, and each element in B is matched with the first choice on its (B-oriented) shortlist, then the resulting matching is stable. This is called the **B-optimal** solution.

- This matching is optimal for B, since no element in B can be matched with a more preferred partner.
- Conversely, no element in A can be matched with a less preferred partner in any other stable matching.

Property 4: B-Optimal Solution

Property 4: If the roles of A and B are interchanged, and each element in B is matched with the first choice on its (B-oriented) shortlist, then the resulting matching is stable. This is called the **B-optimal** solution.

- This matching is optimal for B, since no element in B can be matched with a more preferred partner.
- Conversely, no element in A can be matched with a less preferred partner in any other stable matching.

Key Point: This solution maximizes satisfaction for B, while maintaining stability under the reversed roles.

Shortlists After Gale–Shapley

- The proposer ends up with their best possible stable partner.

Shortlists After Gale–Shapley

- The proposer ends up with their best possible stable partner.
- The receiver is matched with their most preferred option among those who proposed.

Shortlists After Gale–Shapley

- The proposer ends up with their best possible stable partner.
- The receiver is matched with their most preferred option among those who proposed.
- All removed pairs can never appear in any stable matching.

Shortlists After Gale–Shapley

- The proposer ends up with their best possible stable partner.
- The receiver is matched with their most preferred option among those who proposed.
- All removed pairs can never appear in any stable matching.

Conclusion: The mutual shortlists preserve the full set of possible stable matchings.

Singleton Set

After matching, the shortlist can be divided into two sets:

- A set of elements containing only the currently matched partner in the preference list, known as the singleton preference list.
- A set of elements containing more than one element in the preference list.

Rotation

Definition (Rotation)

A **rotation** $p = (a_0, b_0), (a_1, b_1), \ldots, (a_{r-1}, b_{r-1})$ is a sequence of matched pairs where b_i is first in a_i's shortlist, and $b_{(i+1) \bmod r}$ is second in a_i's shortlist.
In the elimination process:

- For each i, the successor x of a_{i-1} in b_i's shortlist is removed.
- The corresponding appearance of b_i is removed from x's preference list.

The rotation p is eliminated when this process is applied for all i, $0 \leq i \leq r-1$, with $i-1$ taken modulo r.

Weight of a Rotation

Given a rotation $\rho = (a_0, b_0), \ldots, (a_{r-1}, b_{r-1})$ in a stable matching instance, we define its weight $w(\rho)$ as:

$$w(\rho) = \sum_{i=0}^{r-1} \left(\mathrm{ar}(a_i, b_i) - \mathrm{ar}(a_i, b_{(i+1) \bmod r}) \right) + \sum_{i=0}^{r-1} \left(\mathrm{br}(b_i, a_i) - \mathrm{br}(b_i, a_{(i-1) \bmod r}) \right)$$

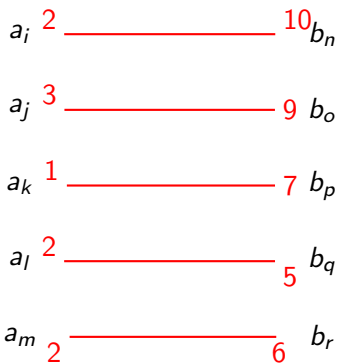
Weight of a Rotation

Given a rotation $\rho = (a_0, b_0), \ldots, (a_{r-1}, b_{r-1})$ in a stable matching instance, we define its weight $w(\rho)$ as:

$$w(\rho) = \sum_{i=0}^{r-1} \left(\mathrm{ar}(a_i, b_i) - \mathrm{ar}(a_i, b_{(i+1) \bmod r}) \right) + \sum_{i=0}^{r-1} \left(\mathrm{br}(b_i, a_i) - \mathrm{br}(b_i, a_{(i-1) \bmod r}) \right)$$

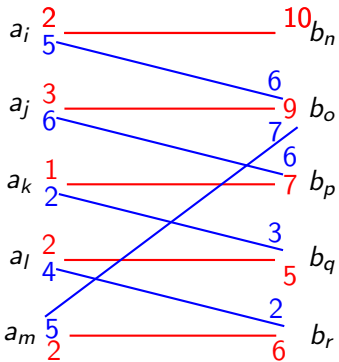
This captures the total change in satisfaction for both sides when rotation ρ is eliminated.

Rotation



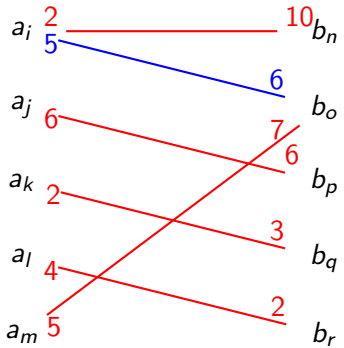
Current matching in some arbitary graph

Rotation



Rotation $(a_j,b_o),(a_k,b_p),(a_l,b_q),(a_m,b_r)$

Rotation



Weight of this rotation: $(3-6)+(1-2)+(2-4)+(5-2)+(9-7)+(7-6)+(5-3)+(6-2) = 6$

Rotation Elimination

A rotation $\rho = (a_0, b_0), (a_1, b_1), \ldots, (a_{r-1}, b_{r-1})$ is said to be **eliminated** if:

- For each i (where $0 \leq i < r$), every successor x of a_{i-1} (modulo r) in b_i's shortlist is removed.
- The corresponding appearance of b_i is also removed from each such x's shortlist.
- These deletions remove all potential for a_{i-1} and b_i to form a better pair in the future.

The rotation ρ is then said to be the **eliminating rotation** for all such removed pairs.

Elimination Prevents Blocking Pairs

Eliminating a rotation ensures stability by preventing blocking pairs:

- A blocking pair (a, b) exists if both prefer each other over their current partners.
- After eliminating a rotation, successors of each a_{i-1} are removed from b_i's list, and vice versa.
- This ensures that neither a_{i-1} nor b_i can later form a more preferred match.
- Therefore, no new blocking pairs can arise from these individuals.

Shortlists After Gale-Shapley

A Preferences

a_1: b_3, b_1, $b5$, ~~b_4~~, ~~b_2~~
a_2: b_1, b_3, b_4, ~~$b5$~~, ~~b_2~~
a_3: b_4, b_3, ~~$b5$~~, b_1, b_2
a_4: $b5$, ~~b_3~~, b_1, ~~b_2~~, b_4
a_5: ~~b_4~~, ~~b_1~~, b_2, b_3, $b5$

B Preferences

b1: a_4, a_3, a_1, a_2, ~~a_5~~
b2: a_3, a_5, ~~a_4~~, ~~a_1~~, ~~a_2~~
b3: a_5, a_3, a_2, a_1, ~~a_4~~
b4: a_4, a_2, a_3, ~~a_1~~, ~~a_5~~
b5: a_1, a_5, a_4, ~~a_3~~, ~~a_2~~

Shortlists After Gale-Shapley

A Preferences

a_1: b_3, b_1, $b5$, ~~b_4~~, ~~b_2~~
a_2: b_1, b_3, b_4, ~~$b5$~~, ~~b_2~~
a_3: b_4, b_3, ~~$b5$~~, b_1, b_2
a_4: $b5$, ~~b_3~~, b_1, ~~b_2~~, b_4
a_5: ~~b_4~~, ~~b_1~~, b_2, b_3, $b5$

B Preferences

b1: a_4, a_3, a_1, a_2, ~~a_5~~
b2: a_3, a_5, ~~a_4~~, ~~a_1~~, ~~a_2~~
b3: a_5, a_3, a_2, a_1, ~~a_4~~
b4: a_4, a_2, a_3, ~~a_1~~, ~~a_5~~
b5: a_1, a_5, a_4, ~~a_3~~, ~~a_2~~

Rotation Identified: $(a_1, b_3), (a_2, b_1)$

Shortlist

Eliminating rotation $(a_1, b_3), (a_2, b_1)$

A Preferences

a_1: ~~b_3~~, b_1, b_5, ~~b_4~~, ~~b_2~~
a_2: ~~b_1~~, b_3, b_4, b_5, ~~b_2~~
a_3: b_4, b_3, ~~b_5~~, b_1, b_2
a_4: b_5, ~~b_3~~, b_1, ~~b_2~~, b_4
a_5: ~~b_4~~, ~~b_1~~, b_2, b_3, b_5

B Preferences

b_1: a_4, a_3, a_1, ~~a_2~~, ~~a_5~~
b_2: a_3, a_5, ~~a_4~~, ~~a_1~~, ~~a_2~~
b_3: a_5, a_3, a_2, ~~a_1~~, ~~a_4~~
b_4: a_4, a_2, a_3, ~~a_1~~, ~~a_5~~
b_5: a_1, a_5, a_4, ~~a_3~~, ~~a_2~~

Shortlist

Eliminating rotation $(a_1, b_3), (a_2, b_1)$

A Preferences

a_1: ~~b_3~~, b_1, b_5, ~~b_4~~, ~~b_2~~
a_2: ~~b_1~~, b_3, b_4, b_5, ~~b_2~~
a_3: b_4, b_3, ~~b_5~~, b_1, b_2
a_4: b_5, ~~b_3~~, b_1, ~~b_2~~, b_4
a_5: ~~b_4~~, ~~b_1~~, b_2, b_3, b_5

B Preferences

b_1: a_4, a_3, a_1, ~~a_2~~, ~~a_5~~
b_2: a_3, a_5, ~~a_4~~, ~~a_1~~, ~~a_2~~
b_3: a_5, a_3, a_2, ~~a_1~~, ~~a_4~~
b_4: a_4, a_2, a_3, ~~a_1~~, ~~a_5~~
b_5: a_1, a_5, a_4, ~~a_3~~, ~~a_2~~

Weight of rotation ρ_1: $(-1 - 1) + (1 + 1) = 0$

Shortlist

Eliminating rotation $(a_1, b_3), (a_2, b_1)$

A Preferences

a_1: ~~b_3~~, b_1, b_5, ~~b_4~~, ~~b_2~~
a_2: ~~b_1~~, b_3, b_4, b_5, ~~b_2~~
a_3: b_4, b_3, ~~b_5~~, b_1, b_2
a_4: b_5, ~~b_3~~, b_1, ~~b_2~~, b_4
a_5: ~~b_4~~, ~~b_1~~, b_2, b_3, b_5

B Preferences

b_1: a_4, a_3, a_1, ~~a_2~~, ~~a_5~~
b_2: a_3, a_5, ~~a_4~~, ~~a_1~~, ~~a_2~~
b_3: a_5, a_3, a_2, ~~a_1~~, ~~a_4~~
b_4: a_4, a_2, a_3, ~~a_1~~, ~~a_5~~
b_5: a_1, a_5, a_4, ~~a_3~~, ~~a_2~~

Weight of rotation ρ_1: $(-1 - 1) + (1 + 1) = 0$
Two new rotations were exposed:

$$(a_1, b_1), (a_4, b_5) \quad \text{and} \quad (a_3, b_4), (a_2, b_3)$$

Shortlist

Eliminating rotation $(a_1, b_3), (a_2, b_1)$

A Preferences

a_1: ~~b_3~~, b_1, b_5, ~~b_4~~, ~~b_2~~
a_2: ~~b_1~~, b_3, b_4, b_5, ~~b_2~~
a_3: b_4, b_3, ~~b_5~~, b_1, b_2
a_4: b_5, ~~b_3~~, b_1, ~~b_2~~, b_4
a_5: ~~b_4~~, ~~b_1~~, b_2, b_3, b_5

B Preferences

b_1: a_4, a_3, a_1, ~~a_2~~, ~~a_5~~
b_2: a_3, a_5, ~~a_4~~, ~~a_1~~, ~~a_2~~
b_3: a_5, a_3, a_2, ~~a_1~~, ~~a_4~~
b_4: a_4, a_2, a_3, ~~a_1~~, ~~a_5~~
b_5: a_1, a_5, a_4, ~~a_3~~, ~~a_2~~

Weight of rotation ρ_1: $(-1-1)+(1+1)=0$
Two new rotations were exposed:

$$(a_1, b_1), (a_4, b_5) \quad \text{and} \quad (a_3, b_4), (a_2, b_3)$$

Note that rotations are not necessarily of size 2.

Shortlist

Eliminating rotation $(a_1, b_1), (a_4, b_5)$

A Preferences

a_1: ~~b_3~~, ~~b_1~~, b_5, ~~b_4~~, ~~b_2~~
a_2: ~~b_1~~, b_3, b_4, ~~b_5~~, ~~b_2~~
a_3: b_4, b_3, ~~b_5~~, ~~b_1~~, b_2
a_4: ~~b_5~~, ~~b_3~~, b_1, ~~b_2~~, b_4
a_5: ~~b_4~~, ~~b_1~~, b_2, b_3, ~~b_5~~

B Preferences

b_1: a_4, ~~a_3~~, ~~a_1~~, a_2, ~~a_5~~
b_2: a_3, a_5, ~~a_4~~, ~~a_1~~, ~~a_2~~
b_3: a_5, a_3, a_2, ~~a_1~~, ~~a_4~~
b_4: a_4, a_2, a_3, ~~a_1~~, ~~a_5~~
b_5: a_1, ~~a_5~~, ~~a_4~~, ~~a_3~~, ~~a_2~~

Shortlist

Eliminating rotation $(a_1, b_1), (a_4, b_5)$

A Preferences

a_1: ~~b_3~~, ~~b_1~~, b_5, ~~b_4~~, ~~b_2~~
a_2: ~~b_1~~, b_3, b_4, ~~b_5~~, ~~b_2~~
a_3: b_4, b_3, ~~b_5~~, ~~b_1~~, b_2
a_4: ~~b_5~~, ~~b_3~~, b_1, ~~b_2~~, b_4
a_5: ~~b_4~~, ~~b_1~~, b_2, b_3, ~~b_5~~

B Preferences

b_1: a_4, ~~a_3~~, ~~a_1~~, ~~a_2~~, ~~a_5~~
b_2: a_3, a_5, ~~a_4~~, ~~a_1~~, ~~a_2~~
b_3: a_5, a_3, a_2, ~~a_1~~, ~~a_4~~
b_4: a_4, a_2, a_3, ~~a_1~~, ~~a_5~~
b_5: a_1, ~~a_5~~, ~~a_4~~, ~~a_3~~, ~~a_2~~

Weight of rotation ρ_2: $(-1-2)+(2+2)=1$

Shortlist

Eliminating rotation $(a_1, b_1), (a_4, b_5)$

A Preferences

a_1: ~~b_3~~, ~~b_1~~, b_5, ~~b_4~~, ~~b_2~~
a_2: ~~b_1~~, b_3, b_4, ~~b_5~~, ~~b_2~~
a_3: b_4, b_3, ~~b_5~~, ~~b_1~~, b_2
a_4: ~~b_5~~, ~~b_3~~, b_1, ~~b_2~~, b_4
a_5: ~~b_4~~, ~~b_1~~, b_2, b_3, ~~b_5~~

B Preferences

b_1: a_4, ~~a_3~~, ~~a_1~~, ~~a_2~~, ~~a_5~~
b_2: a_3, a_5, ~~a_4~~, ~~a_1~~, ~~a_2~~
b_3: a_5, a_3, a_2, ~~a_1~~, ~~a_4~~
b_4: a_4, a_2, a_3, ~~a_1~~, ~~a_5~~
b_5: a_1, ~~a_5~~, ~~a_4~~, ~~a_3~~, ~~a_2~~

Weight of rotation ρ_2: $(-1-2)+(2+2)=1$
No new rotation was exposed.

Shortlist

Eliminating rotation $(a_1, b_1), (a_4, b_5)$

A Preferences

a_1: ~~b_3~~, ~~b_1~~, b_5, ~~b_4~~, ~~b_2~~
a_2: ~~b_1~~, b_3, b_4, ~~b_5~~, ~~b_2~~
a_3: b_4, b_3, ~~b_5~~, ~~b_1~~, b_2
a_4: ~~b_5~~, ~~b_3~~, b_1, ~~b_2~~, b_4
a_5: ~~b_4~~, ~~b_1~~, b_2, b_3, ~~b_5~~

B Preferences

b_1: a_4, ~~a_3~~, ~~a_1~~, ~~a_2~~, ~~a_5~~
b_2: a_3, a_5, ~~a_4~~, ~~a_1~~, ~~a_2~~
b_3: a_5, a_3, a_2, ~~a_1~~, ~~a_4~~
b_4: a_4, a_2, a_3, ~~a_1~~, ~~a_5~~
b_5: a_1, ~~a_5~~, ~~a_4~~, ~~a_3~~, ~~a_2~~

Weight of rotation ρ_2: $(-1-2)+(2+2)=1$
No new rotation was exposed.
Now we observe that a_1,b_1 and b_5 have been removed from everyone other than their matched partner.
Hence, no future rotations can involve these.

Shortlist

Eliminating rotation $(a_2, b_3), (a_3, b_4)$

A Preferences

a_1: ~~b_3, b_1~~, b_5, ~~b_4, b_2~~
a_2: ~~b_1~~, b_3, b_4, ~~b_5, b_2~~
a_3: ~~b_4~~, b_3, ~~b_5, b_1~~, b_2
a_4: ~~b_5, b_3~~, b_1, ~~b_2~~, b_4
a_5: ~~b_4, b_1~~, b_2, b_3, ~~b_5~~

B Preferences

b_1: a_4, ~~a_3, a_1, a_2, a_5~~
b_2: a_3, a_5, ~~a_4, a_1, a_2~~
b_3: a_5, a_3, ~~a_2, a_1, a_4~~
b_4: a_4, a_2, ~~a_3, a_1, a_5~~
b_5: a_1, ~~a_5, a_4, a_3, a_2~~

Shortlist

Eliminating rotation $(a_2, b_3), (a_3, b_4)$

A Preferences

a_1: ~~b_3, b_1~~, b_5, ~~b_4, b_2~~
a_2: ~~b_1~~, ~~b_3~~, b_4, ~~b_5, b_2~~
a_3: ~~b_4~~, b_3, ~~b_5, b_1~~, b_2
a_4: ~~b_5, b_3~~, b_1, ~~b_2~~, b_4
a_5: ~~b_4, b_1~~, b_2, b_3, ~~b_5~~

B Preferences

b_1: a_4, ~~a_3, a_1, a_2, a_5~~
b_2: a_3, a_5, ~~a_4, a_1, a_2~~
b_3: a_5, a_3, ~~a_2, a_1, a_4~~
b_4: a_4, a_2, ~~a_3, a_1, a_5~~
b_5: a_1, ~~a_5, a_4, a_3, a_2~~

Weight of rotation ρ_3: $(-1-1)+(1+1)=0$

Shortlist

Eliminating rotation $(a_2, b_3), (a_3, b_4)$

A Preferences

a_1: ~~b_3, b_1~~, b_5, ~~b_4, b_2~~
a_2: ~~b_1, b_3~~, b_4, ~~b_5, b_2~~
a_3: ~~b_4~~, b_3, ~~b_5, b_1~~, b_2
a_4: ~~b_5, b_3~~, b_1, ~~b_2~~, b_4
a_5: ~~b_4, b_1~~, b_2, b_3, ~~b_5~~

B Preferences

b_1: a_4, ~~a_3, a_1, a_2, a_5~~
b_2: a_3, a_5, ~~a_4, a_1, a_2~~
b_3: a_5, a_3, ~~a_2, a_1, a_4~~
b_4: a_4, a_2, ~~a_3, a_1, a_5~~
b_5: a_1, ~~a_5, a_4, a_3, a_2~~

Weight of rotation ρ_3: $(-1 - 1) + (1 + 1) = 0$
New rotation exposed:

$$(a_3, b_3), (a_5, b_2)$$

Shortlist

Eliminating rotation $(a_3, b_3), (a_5, b_2)$

A Preferences

a_1: ~~b_3~~, ~~b_1~~, b_5, ~~b_4~~, ~~b_2~~
a_2: ~~b_1~~, ~~b_3~~, b_4, ~~b_5~~, ~~b_2~~
a_3: ~~b_4~~, ~~b_3~~, ~~b_5~~, ~~b_1~~, b_2
a_4: ~~b_5~~, ~~b_3~~, b_1, ~~b_2~~, b_4
a_5: ~~b_4~~, ~~b_1~~, ~~b_2~~, b_3, ~~b_5~~

B Preferences

b_1: a_4, ~~a_3~~, ~~a_1~~, ~~a_2~~, ~~a_5~~
b_2: a_3, ~~a_5~~, ~~a_4~~, ~~a_1~~, ~~a_2~~
b_3: a_5, ~~a_3~~, ~~a_2~~, ~~a_1~~, ~~a_4~~
b_4: a_4, a_2, ~~a_3~~, ~~a_1~~, ~~a_5~~
b_5: a_1, ~~a_5~~, ~~a_4~~, ~~a_3~~, ~~a_2~~

Shortlist

Eliminating rotation $(a_3, b_3), (a_5, b_2)$

A Preferences

a_1: ~~b_3~~, ~~b_1~~, b_5, ~~b_4~~, ~~b_2~~
a_2: ~~b_1~~, ~~b_3~~, b_4, ~~b_5~~, ~~b_2~~
a_3: ~~b_4~~, b_3, ~~b_5~~, ~~b_1~~, b_2
a_4: ~~b_5~~, ~~b_3~~, b_1, ~~b_2~~, b_4
a_5: ~~b_4~~, ~~b_1~~, b_2, b_3, ~~b_5~~

B Preferences

b_1: a_4, ~~a_3~~, ~~a_1~~, ~~a_2~~, ~~a_5~~
b_2: a_3, ~~a_5~~, ~~a_4~~, ~~a_1~~, ~~a_2~~
b_3: a_5, ~~a_3~~, ~~a_2~~, ~~a_1~~, ~~a_4~~
b_4: a_4, a_2, ~~a_3~~, ~~a_1~~, ~~a_5~~
b_5: a_1, ~~a_5~~, ~~a_4~~, ~~a_3~~, ~~a_2~~

Weight of the rotation ρ_4: $(-3-1)+(1+1)=-2$

Shortlist

Eliminating rotation $(a_3, b_3), (a_5, b_2)$

A Preferences

a_1: ~~b_3~~, ~~b_1~~, b_5, ~~b_4~~, ~~b_2~~
a_2: ~~b_1~~, ~~b_3~~, b_4, ~~b_5~~, ~~b_2~~
a_3: ~~b_4~~, ~~b_3~~, ~~b_5~~, ~~b_1~~, b_2
a_4: ~~b_5~~, ~~b_3~~, b_1, ~~b_2~~, b_4
a_5: ~~b_4~~, ~~b_1~~, ~~b_2~~, b_3, ~~b_5~~

B Preferences

b_1: a_4, ~~a_3~~, ~~a_1~~, ~~a_2~~, ~~a_5~~
b_2: a_3, ~~a_5~~, ~~a_4~~, ~~a_1~~, ~~a_2~~
b_3: a_5, ~~a_3~~, ~~a_2~~, ~~a_1~~, ~~a_4~~
b_4: a_4, a_2, ~~a_3~~, ~~a_1~~, ~~a_5~~
b_5: a_1, ~~a_5~~, ~~a_4~~, ~~a_3~~, ~~a_2~~

Weight of the rotation ρ_4: $(-3-1)+(1+1)=-2$
Reached B optimal matching.

Shortlist

Eliminating rotation $(a_3, b_3), (a_5, b_2)$

A Preferences

a_1: ~~b_3~~, ~~b_1~~, b_5, ~~b_4~~, ~~b_2~~
a_2: ~~b_1~~, ~~b_3~~, b_4, ~~b_5~~, ~~b_2~~
a_3: ~~b_4~~, b_3, ~~b_5~~, ~~b_1~~, b_2
a_4: ~~b_5~~, ~~b_3~~, b_1, ~~b_2~~, b_4
a_5: ~~b_4~~, ~~b_1~~, b_2, b_3, ~~b_5~~

B Preferences

b_1: a_4, ~~a_3~~, ~~a_1~~, ~~a_2~~, ~~a_5~~
b_2: a_3, ~~a_5~~, ~~a_4~~, ~~a_1~~, ~~a_2~~
b_3: a_5, ~~a_3~~, ~~a_2~~, ~~a_1~~, ~~a_4~~
b_4: a_4, a_2, ~~a_3~~, ~~a_1~~, ~~a_5~~
b_5: a_1, ~~a_5~~, ~~a_4~~, ~~a_3~~, ~~a_2~~

Weight of the rotation ρ_4: $(-3-1)+(1+1)=-2$
Reached B optimal matching.
No more rotations possible.

Property

Theorem

Every stable matching can be obtained by starting from the shortlists and eliminating some sequence of zero or more exposed rotations.

Number of Rotations

Although a stable marriage instance with n A and n B can have exponentially many stable matchings, the number of **rotations** is bounded by n^2.

Number of Rotations

Although a stable marriage instance with n A and n B can have exponentially many stable matchings, the number of **rotations** is bounded by n^2.

This is because each pair (a, b) can appear in at most one rotation, and there are only n^2 such possible pairs in total.
Also no pair is eliminated more than once

Explicit Predecessor and the Rotation Poset

A rotation π is called an **explicit predecessor** of the rotation

$$\rho = (a_0, b_0), (a_1, b_1), \ldots, (a_{r-1}, b_{r-1})$$

if, for some index $i \in [0, r-1]$ and some $x \in B$ with $x \neq b_i$:

- The rotation π is the *eliminating rotation* for the pair (a_i, x), and
- a_i prefers x over b_{i+1}.

Explicit Predecessor and the Rotation Poset

A rotation π is called an **explicit predecessor** of the rotation

$$\rho = (a_0, b_0), (a_1, b_1), \ldots, (a_{r-1}, b_{r-1})$$

if, for some index $i \in [0, r-1]$ and some $x \in B$ with $x \neq b_i$:

- The rotation π is the *eliminating rotation* for the pair (a_i, x), and
- a_i prefers x over b_{i+1}.

The **rotation poset** for a problem instance is a partially ordered set where:

$$\alpha < \rho \iff \alpha \text{ must be eliminated before } \rho \text{ becomes exposed.}$$

A subset $C \subseteq P$, where $P = (I, \leq)$ is a poset, is called a *closed set* if:

$$\rho \in C,\ \pi < \rho \Rightarrow \pi \in C.$$

That is, all predecessors of any element in C must also lie in C.

Rotation Poset

π is an immediate predecessor of ρ if

$$\pi < \rho \quad \text{and there is no } \sigma \text{ such that } \pi < \sigma < \rho.$$

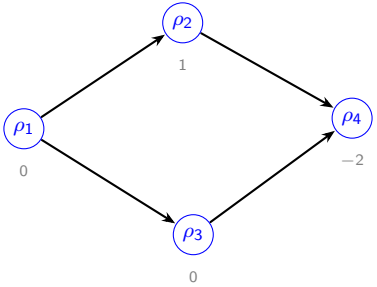
Rotation Poset

π is an immediate predecessor of ρ if

$$\pi < \rho \quad \text{and there is no } \sigma \text{ such that } \pi < \sigma < \rho.$$

Rotation	**Immediate Predecessors**
$\rho_1 = (a1, b3),\ (a2, b1)$	—
$\rho_2 = (a1, b1),\ (a4, b5)$	ρ_1
$\rho_3 = (a2, b3),\ (a3, b4)$	ρ_1
$\rho_4 = (a3, b3),\ (a5, b2)$	$\rho_2,\ \rho_3$

Rotation Poset



Why always DAG?

A **rotation poset** captures dependencies between rotations: $\rho_1 < \rho_2$ means ρ_1 must be eliminated before ρ_2 becomes exposed.

Why always DAG?

A **rotation poset** captures dependencies between rotations: $\rho_1 < \rho_2$ means ρ_1 must be eliminated before ρ_2 becomes exposed.

Why is it always a DAG?

- The "$<$" relation is a strict partial order.

Why always DAG?

A **rotation poset** captures dependencies between rotations: $\rho_1 < \rho_2$ means ρ_1 must be eliminated before ρ_2 becomes exposed.

Why is it always a DAG?

- The "$<$" relation is a strict partial order.
- A cycle would imply a set of rotations must all come before each other $\Rightarrow$ contradiction.

Why always DAG?

A **rotation poset** captures dependencies between rotations: $\rho_1 < \rho_2$ means ρ_1 must be eliminated before ρ_2 becomes exposed.

Why is it always a DAG?

- The "$<$" relation is a strict partial order.
- A cycle would imply a set of rotations must all come before each other $\Rightarrow$ contradiction.
- In stable matching, each eliminated rotation strictly reduces the possible matchings.

Why always DAG?

A **rotation poset** captures dependencies between rotations: $\rho_1 < \rho_2$ means ρ_1 must be eliminated before ρ_2 becomes exposed.

Why is it always a DAG?

- The "$<$" relation is a strict partial order.
- A cycle would imply a set of rotations must all come before each other $\Rightarrow$ contradiction.
- In stable matching, each eliminated rotation strictly reduces the possible matchings.
- After eliminating a rotation:
  - Each A removes all partners **before** the newly matched one in her list.
  - Each B removes all partners **after** the newly matched one in his list.

Why always DAG?

A **rotation poset** captures dependencies between rotations: $\rho_1 < \rho_2$ means ρ_1 must be eliminated before ρ_2 becomes exposed.

Why is it always a DAG?

- The "$<$" relation is a strict partial order.
- A cycle would imply a set of rotations must all come before each other $\Rightarrow$ contradiction.
- In stable matching, each eliminated rotation strictly reduces the possible matchings.
- After eliminating a rotation:
  - Each A removes all partners **before** the newly matched one in her list.
  - Each B removes all partners **after** the newly matched one in his list.
- This pruning ensures no earlier rotation can become exposed again.

Why always DAG?

A **rotation poset** captures dependencies between rotations: $\rho_1 < \rho_2$ means ρ_1 must be eliminated before ρ_2 becomes exposed.

Why is it always a DAG?

- The "$<$" relation is a strict partial order.
- A cycle would imply a set of rotations must all come before each other $\Rightarrow$ contradiction.
- In stable matching, each eliminated rotation strictly reduces the possible matchings.
- After eliminating a rotation:
  - Each A removes all partners **before** the newly matched one in her list.
  - Each B removes all partners **after** the newly matched one in his list.
- This pruning ensures no earlier rotation can become exposed again.
- $\Rightarrow$ No cyclic dependencies possible $\Rightarrow$ Always a DAG.

Corollary

Let S be the stable matching obtained by eliminating the sequence of rotations $\rho_1, \ldots, \rho_k$ from the shortlists of a stable marriage instance.

Corollary

Let S be the stable matching obtained by eliminating the sequence of rotations $\rho_1, \ldots, \rho_k$ from the shortlists of a stable marriage instance.
Then,

$$c(S) = c(S_0) - \sum_{i=1}^{k} w(\rho_i)$$

where $c(S_0)$ is the cost of the initial stable matching (e.g., A-optimal).

Corollary

Let S be the stable matching obtained by eliminating the sequence of rotations $\rho_1, \ldots, \rho_k$ from the shortlists of a stable marriage instance.
Then,

$$c(S) = c(S_0) - \sum_{i=1}^{k} w(\rho_i)$$

where $c(S_0)$ is the cost of the initial stable matching (e.g., A-optimal).
Therefore, to find an optimal stable matching, it suffices to construct a closed subset of the rotation poset with the **maximum total weight**.

Corollary

Let S be the stable matching obtained by eliminating the sequence of rotations $\rho_1, \ldots, \rho_k$ from the shortlists of a stable marriage instance.
Then,

$$c(S) = c(S_0) - \sum_{i=1}^{k} w(\rho_i)$$

where $c(S_0)$ is the cost of the initial stable matching (e.g., A-optimal).
Therefore, to find an optimal stable matching, it suffices to construct a closed subset of the rotation poset with the **maximum total weight**.

Example

The set $\{\rho_1, \rho_2\}$ forms a closed subset with total weight $+1$.

Finding maximum weight closed subset of P

We construct an s–t flow graph on the poset P as follows:

- Add a source node s and a sink node t to the graph.
- For every **negative-weight** node ρ_i, add a directed edge from s to ρ_i with capacity:

$$\text{capacity}(s, \rho_i) = |w(\rho_i)|$$

- For every **positive-weight** node ρ_j, add a directed edge from ρ_j to t with capacity:

$$\text{capacity}(\rho_j, t) = w(\rho_j)$$

- Set the capacity of every original edge in P (i.e., edges that represent ordering constraints) to ∞.

Theorem: Closed Subsets via Min-Cut

Theorem

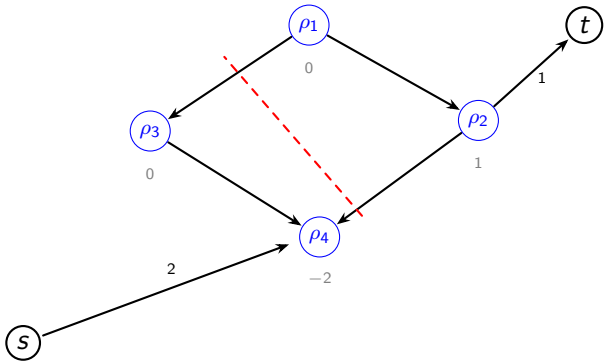
Let X be the set of edges crossing a minimum s-t cut in the flow network $P(s,t)$, with total capacity $w(X)$.

Then, the positive nodes in the maximum-weight closed subset of P' are:

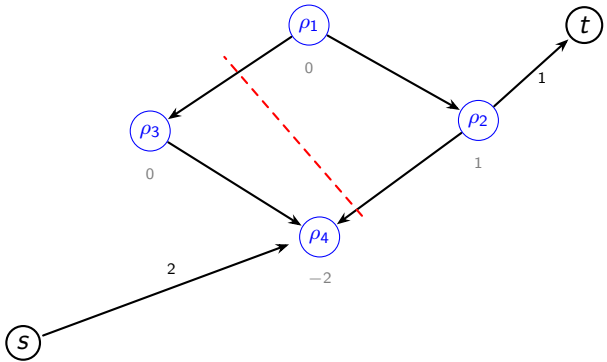
- *Exactly those positive nodes whose edges into t are* **not** *in the cut X,*
- *Along with all nodes that can reach them in P' (i.e., their predecessors).*

These nodes collectively form a **maximum-weight closed subset** *in P.*

Rotation Poset

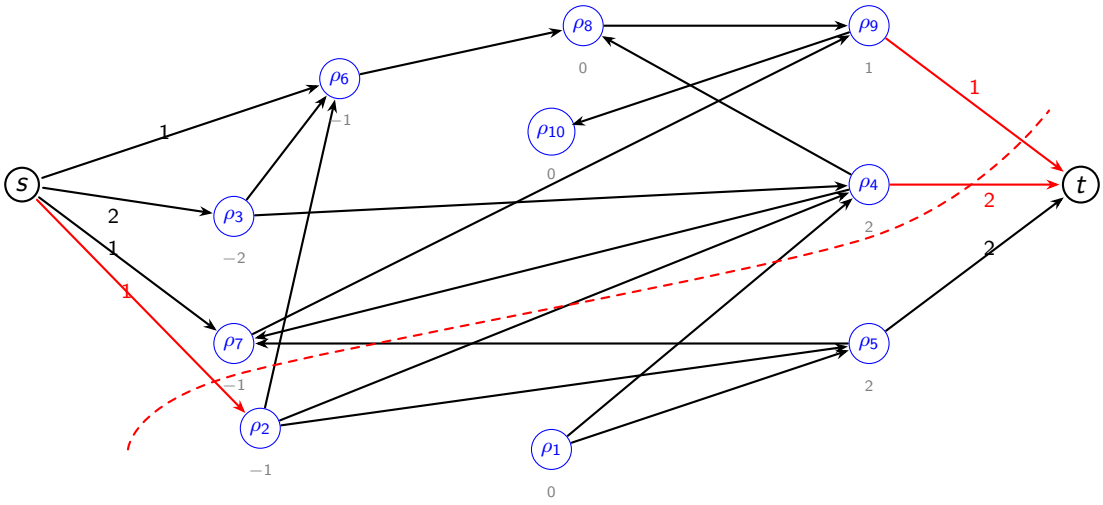


Rotation Poset



poset with sum$\rho_1, \rho_2 = 1$

Always cut through ∞ weight edges edges?



Poset (ρ_1, ρ_2, ρ_5) with sum $+1$

Complexity Analysis

- **Finding all rotations:** $O(n^3)$
- **Finding minimum s-t cut:** $O(n^4)$
- **Maximum-weight closed subset via cut:** $O(n^2)$
- **Rotation elimination:** $O(n^2)$

Note: Any improvement in flow computation directly improves the overall bound.

Overall complexity: $\boxed{O(n^4)}$