Computational Social Choice and Complexity Theory

Ronald de Haan
University of Amsterdam

https://staff.science.uva.nl/r.dehaan/esslli2018
me@ronalddehaan.eu

ESSLLI 2018 – Day 5
Recap

- Voting
- Winner Determination, Manipulation, Bribery
- Domain Restrictions, Single-Peakedness
- Judgment Aggregation
What we’ll do today

▶ Stable Matching
▶ The Gale-Shapley Algorithm
▶ Strategizing
▶ Variants of Matching Problems
Stable Matching
Bipartite Matching of Agents

\[ a_1 \bullet \quad \bullet \quad b_1 \]

\[ a_2 \bullet \quad \bullet \quad b_2 \]

preference of \( a_1 \):

\[ b_1 \succ_{a_1} b_2 \]

preference of \( a_2 \):

\[ b_2 \succ_{a_2} b_1 \]

preference of \( b_1 \):

\[ a_2 \succ_{b_1} a_1 \]

preference of \( b_2 \):

\[ a_2 \succ_{b_2} a_1 \]
Bipartite Matching of Agents

\[ a_1 \sim b_1 \]

\[ a_2 \sim b_2 \]

preference of \( a_1 \): \( b_1 \succ a_1 b_2 \)
preference of \( a_2 \): \( b_2 \succ a_2 b_1 \)
preference of \( b_1 \): \( a_2 \succ b_1 a_1 \)
preference of \( b_2 \): \( a_2 \succ b_2 a_1 \)
Bipartite Matching of Agents

preference of $a_1$: $b_1 \succ a_1 b_2$

preference of $a_2$: $b_2 \succ a_2 b_1$

preference of $b_1$: $a_2 \succ b_1 a_1$

preference of $b_2$: $a_2 \succ b_2 a_1$
Stable matching

- Two sets of agents (of same size):
  - $A = \{a_1, \ldots, a_n\}$
  - $B = \{b_1, \ldots, b_n\}$

- Each agent has a preference over all agents from the other side (candidates): a linear order $\succ$
  - All preferences together: preference profile

- A matching is a bijection $\mu$ between $A$ and $B$
  - Blocking pair: $(a, b) \in A \times B$ such that:
    - $(a, b) \notin \mu$,
    - $b \succ_a \mu(a)$, and
    - $a \succ_b \mu(b)$.
  - Matching $\mu$ is stable if there exists no blocking pair
Example

\[ A = \{a_1, a_2\}, \quad B = \{b_1, b_2\} \]

preference of \( a_1 \): \( b_1 \succ a_1 b_2 \)
preference of \( a_2 \): \( b_2 \succ a_2 b_1 \)

preference of \( b_1 \): \( a_2 \succ b_1 a_1 \)
preference of \( b_2 \): \( a_2 \succ b_2 a_1 \)
Example

Stable matching:

\[ a_1 \quad \bullet \quad \bullet \quad b_1 \]

\[ a_2 \quad \bullet \quad \bullet \quad b_2 \]

\[ A = \{a_1, a_2\}, \quad B = \{b_1, b_2\} \]

preference of \( a_1 \): \( b_1 \succ a_1 b_2 \)

preference of \( a_2 \): \( b_2 \succ a_2 b_1 \)

preference of \( b_1 \): \( a_2 \succ b_1 a_1 \)

preference of \( b_2 \): \( a_2 \succ b_2 a_1 \)
Example

Unstable matching:

\[ A = \{a_1, a_2\}, \quad B = \{b_1, b_2\} \]

preference of \( a_1 \): \( b_1 \succ a_1 b_2 \)
preference of \( a_2 \): \( b_2 \succ a_2 b_1 \)

preference of \( b_1 \): \( a_2 \succ b_1 a_1 \)
preference of \( b_2 \): \( a_2 \succ b_2 a_1 \)
Example

Unstable matching:

\[ A = \{ a_1, a_2 \}, \quad B = \{ b_1, b_2 \} \]

- preference of \( a_1 \): \( b_1 \succ a_1 b_2 \)
- preference of \( a_2 \): \( b_2 \succ a_2 b_1 \)
- preference of \( b_1 \): \( a_2 \succ b_1 a_1 \)
- preference of \( b_2 \): \( a_2 \succ b_2 a_1 \)
The Gale-Shapley Algorithm

- Does a stable matching always exist?
- Can we find a stable matching efficiently, if it exists?

**Answers:** yes, and yes.

- A stable matching always exists and we can use the Gale-Shapley algorithm to find one.

The Gale-Shapley Algorithm

- We choose one side (say $A$) as proposing side, and we construct the matching $\mu$ in rounds.

- In each round, a currently unmatched agent $a \in A$ proposes to their top ranked agent $b \in B$ that they have not proposed to before.

- When some $b \in B$ is proposed to by $a \in A$:
  - if $b$ is currently unmatched, they provisionally accept the match with $a$.
  - if $b$ is currently matched to $a'$, and $a' \succ_b a$, then $b$ rejects $a$.
  - if $b$ is currently matched to $a'$, and $a \succ_b a'$, then $b$ rejects their previous match $a'$ (and $a'$ becomes unmatched again).

- We continue until all agents are matched, and return the constructed matching $\mu$. 
The Gale-Shapley Algorithm (Example Run 1)

preference of $a_1$: $b_1 ∼_{a_1} b_2 ∼_{a_1} b_3$
preference of $a_2$: $b_1 ∼_{a_2} b_2 ∼_{a_2} b_3$
preference of $a_3$: $b_2 ∼_{a_3} b_1 ∼_{a_3} b_3$
preference of $b_1$: $a_3 ∼_{b_1} a_2 ∼_{b_1} a_1$
preference of $b_2$: $a_2 ∼_{b_2} a_3 ∼_{b_2} a_1$
preference of $b_3$: $a_2 ∼_{b_3} a_3 ∼_{b_3} a_1
The Gale-Shapley Algorithm (Example Run 1)

preference of $a_1$: $b_1 \succ a_1 \ b_2 \succ a_1 \ b_3$

preference of $a_2$: $b_1 \succ a_2 \ b_2 \succ a_2 \ b_3$

preference of $a_3$: $b_2 \succ a_3 \ b_1 \succ a_3 \ b_3$

preference of $b_1$: $a_3 \succ b_1 \ a_2 \succ b_1 \ a_1$

preference of $b_2$: $a_2 \succ b_2 \ a_3 \succ b_2 \ a_1$

preference of $b_3$: $a_2 \succ b_3 \ a_3 \succ b_3 \ a_1$
The Gale-Shapley Algorithm (Example Run 1)

preference of $a_1$: \[ b_1 \succ a_1 \succ b_2 \succ a_1 \succ b_3 \]
preference of $a_2$: \[ b_1 \succ a_2 \succ b_2 \succ a_2 \succ b_3 \]
preference of $a_3$: \[ b_2 \succ a_3 \succ b_1 \succ a_3 \succ b_3 \]

preference of $b_1$: \[ a_3 \succ b_1 \succ a_2 \succ b_1 \succ a_1 \]
preference of $b_2$: \[ a_2 \succ b_2 \succ a_3 \succ b_2 \succ a_1 \]
preference of $b_3$: \[ a_2 \succ b_3 \succ a_3 \succ b_3 \succ a_1 \]
The Gale-Shapley Algorithm (Example Run 1)

preference of $a_1$: $b_1 \succ_a b_2 \succ_a b_3$
preference of $a_2$: $b_1 \succ_a b_2 \succ_a b_3$
preference of $a_3$: $b_2 \succ_a b_1 \succ_a b_3$

preference of $b_1$: $a_3 \succ b_1 a_2 \succ b_1 a_1$
preference of $b_2$: $a_2 \succ b_2 a_3 \succ b_2 a_1$
preference of $b_3$: $a_2 \succ b_3 a_3 \succ b_3 a_1$
The Gale-Shapley Algorithm (Example Run 1)

preference of $a_1$: $b_1 \succ_a b_2 \succ_a b_3$
preference of $a_2$: $b_1 \succ_a b_2 \succ_a b_3$
preference of $a_3$: $b_2 \succ_a b_1 \succ_a b_3$

preference of $b_1$: $a_3 \succ_b a_2 \succ_b a_1$
preference of $b_2$: $a_2 \succ_b a_3 \succ_b a_1$
preference of $b_3$: $a_2 \succ_b a_3 \succ_b a_1$
The Gale-Shapley Algorithm (Example Run 1)

preference of $a_1$: $b_1 \succ_{a_1} b_2 \succ_{a_1} b_3$
preference of $a_2$: $b_1 \succ_{a_2} b_2 \succ_{a_2} b_3$
preference of $a_3$: $b_2 \succ_{a_3} b_1 \succ_{a_3} b_3$
preference of $b_1$: $a_3 \succ b_1 \ a_2 \succ b_1 \ a_1$
preference of $b_2$: $a_2 \succ b_2 \ a_3 \succ b_2 \ a_1$
preference of $b_3$: $a_2 \succ b_3 \ a_3 \succ b_3 \ a_1$
The Gale-Shapley Algorithm (Example Run 1)

preference of $a_1$: $b_1 \succ_a b_2 \succ_a b_3$

preference of $a_2$: $b_1 \succ_a b_2 \succ_a b_3$

preference of $a_3$: $b_2 \succ_a b_1 \succ_a b_3$

preference of $b_1$: $a_3 \succ_b a_2 \succ_b a_1$

preference of $b_2$: $a_2 \succ_b a_3 \succ_b a_1$

preference of $b_3$: $a_2 \succ_b a_3 \succ_b a_1$
The Gale-Shapley Algorithm (Example Run 1)

preference of $a_1$: $b_1 \succ a_1 \ b_2 \succ a_1 \ b_3$
preference of $a_2$: $b_1 \succ a_2 \ b_2 \succ a_2 \ b_3$
preference of $a_3$: $b_2 \succ a_3 \ b_1 \succ a_3 \ b_3$
preference of $b_1$: $a_3 \succ b_1 \ a_2 \succ b_1 \ a_1$
preference of $b_2$: $a_2 \succ b_2 \ a_3 \succ b_2 \ a_1$
preference of $b_3$: $a_2 \succ b_3 \ a_3 \succ b_3 \ a_1$
The Gale-Shapley Algorithm (Example Run 1)

preference of $a_1$: $b_1 \succ_a b_2 \succ_a b_3$
preference of $a_2$: $b_1 \succ_a b_2 \succ_a b_3$
preference of $a_3$: $b_2 \succ_a b_1 \succ_a b_3$
preference of $b_1$: $a_3 \succ b_1 a_2 \succ b_1 a_1$
preference of $b_2$: $a_2 \succ b_2 a_3 \succ b_2 a_1$
preference of $b_3$: $a_2 \succ b_3 a_3 \succ b_3 a_1$
The Gale-Shapley Algorithm (Example Run 2)

- preference of $a_1$: $b_3 \succ a_1$, $b_1 \succ a_1$, $b_2 \succ a_1$
- preference of $a_2$: $b_3 \succ a_2$, $b_2 \succ a_2$, $b_1 \succ a_2$
- preference of $a_3$: $b_2 \succ a_3$, $b_1 \succ a_3$, $b_3 \succ a_3$
- preference of $b_1$: $a_3 \succ b_1$, $a_2 \succ b_1$, $a_1 \succ b_1$
- preference of $b_2$: $a_3 \succ b_2$, $a_2 \succ b_2$, $a_1 \succ b_2$
- preference of $b_3$: $a_2 \succ b_3$, $a_3 \succ b_3$, $a_1 \succ b_3$
The Gale-Shapley Algorithm (Example Run 2)

preference of $a_1$: $b_1 \succ_{a_1} b_2 \succ_{a_1} b_3$
preference of $a_2$: $b_1 \succ_{a_2} b_2 \succ_{a_2} b_3$
preference of $a_3$: $b_2 \succ_{a_3} b_1 \succ_{a_3} b_3$

preference of $b_1$: $a_3 \succ b_1 a_2 \succ b_1 a_1$
preference of $b_2$: $a_2 \succ b_2 a_3 \succ b_2 a_1$
preference of $b_3$: $a_2 \succ b_3 a_3 \succ b_3 a_1$
The Gale-Shapley Algorithm (Example Run 2)

preference of $a_1$: $b_1 \succ_{a_1} b_2 \succ_{a_1} b_3$
preference of $a_2$: $b_1 \succ_{a_2} b_2 \succ_{a_2} b_3$
preference of $a_3$: $b_2 \succ_{a_3} b_1 \succ_{a_3} b_3$

preference of $b_1$: $a_3 \succ_{b_1} a_2 \succ_{b_1} a_1$
preference of $b_2$: $a_2 \succ_{b_2} a_3 \succ_{b_2} a_1$
preference of $b_3$: $a_2 \succ_{b_3} a_3 \succ_{b_3} a_1$
The Gale-Shapley Algorithm (Example Run 2)

preference of $a_1$: $b_1 \succ_a b_2 \succ_a b_3$
preference of $a_2$: $b_1 \succ_a b_2 \succ_a b_3$
preference of $a_3$: $b_2 \succ_a b_1 \succ_a b_3$

preference of $b_1$: $a_3 \succ_b a_2 \succ_b a_1$
preference of $b_2$: $a_2 \succ_b a_3 \succ_b a_1$
preference of $b_3$: $a_2 \succ_b a_3 \succ_b a_1$
The Gale-Shapley Algorithm (Example Run 2)

preference of $a_1$: $b_1 \succ_{a_1} b_2 \succ_{a_1} b_3$

preference of $a_2$: $b_1 \succ_{a_2} b_2 \succ_{a_2} b_3$

preference of $a_3$: $b_2 \succ_{a_3} b_1 \succ_{a_3} b_3$

preference of $b_1$: $a_3 \succ b_1 a_2 \succ b_1 a_1$

preference of $b_2$: $a_2 \succ b_2 a_3 \succ b_2 a_1$

preference of $b_3$: $a_2 \succ b_3 a_3 \succ b_3 a_1$
The Gale-Shapley Algorithm (Example Run 2)

preference of $a_1$: $b_1 \succ a_1 b_2 \succ a_1 b_3$
preference of $a_2$: $b_1 \succ a_2 b_2 \succ a_2 b_3$
preference of $a_3$: $b_2 \succ a_3 b_1 \succ a_3 b_3$

preference of $b_1$: $a_3 \succ b_1 a_2 \succ b_1 a_1$
preference of $b_2$: $a_2 \succ b_2 a_3 \succ b_2 a_1$
preference of $b_3$: $a_2 \succ b_3 a_3 \succ b_3 a_1$
The Gale-Shapley Algorithm (Example Run 2)

preference of $a_1$: $b_1 \succ_{a_1} b_2 \succ_{a_1} b_3$

preference of $a_2$: $b_1 \succ_{a_2} b_2 \succ_{a_2} b_3$

preference of $a_3$: $b_2 \succ_{a_3} b_1 \succ_{a_3} b_3$

preference of $b_1$: $a_3 \succ_{b_1} a_2 \succ_{b_1} a_1$

preference of $b_2$: $a_2 \succ_{b_2} a_3 \succ_{b_2} a_1$

preference of $b_3$: $a_2 \succ_{b_3} a_3 \succ_{b_3} a_1$
The Gale-Shapley Algorithm (Example Run 2)

preference of $a_1$: $b_1 \succ a_1 b_2 \succ a_1 b_3$
preference of $a_2$: $b_1 \succ a_2 b_2 \succ a_2 b_3$
preference of $a_3$: $b_2 \succ a_3 b_1 \succ a_3 b_3$

preference of $b_1$: $a_3 \succ b_1 a_2 \succ b_1 a_1$
preference of $b_2$: $a_2 \succ b_2 a_3 \succ b_2 a_1$
preference of $b_3$: $a_2 \succ b_3 a_3 \succ b_3 a_1$
The Gale-Shapley Algorithm (Example Run 2)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>$b_1$</td>
<td></td>
</tr>
<tr>
<td>$a_2$</td>
<td>$b_2$</td>
<td></td>
</tr>
<tr>
<td>$a_3$</td>
<td>$b_3$</td>
<td></td>
</tr>
</tbody>
</table>

preference of $a_1$: $b_1 \succ a_1 b_2 \succ a_1 b_3$
preference of $a_2$: $b_1 \succ a_2 b_2 \succ a_2 b_3$
preference of $a_3$: $b_2 \succ a_3 b_1 \succ a_3 b_3$

preference of $b_1$: $a_3 \succ b_1 a_2 \succ b_1 a_1$
preference of $b_2$: $a_2 \succ b_2 a_3 \succ b_2 a_1$
preference of $b_3$: $a_2 \succ b_3 a_3 \succ b_3 a_1$
The Gale-Shapley Algorithm (Example Run 3)

\[a_1 \bullet \quad \bullet \ b_1\]
\[a_2 \bullet \quad \bullet \ b_2\]
\[a_3 \bullet \quad \bullet \ b_3\]

preference of \(a_1\): \(b_1 \succ a_1 \succ b_2 \succ a_1 \succ b_3\)
preference of \(a_2\): \(b_1 \succ a_2 \succ b_2 \succ a_2 \succ b_3\)
preference of \(a_3\): \(b_2 \succ a_3 \succ b_1 \succ a_3 \succ b_3\)

preference of \(b_1\): \(a_3 \succ b_1 \succ a_2 \succ b_1 \succ a_1\)
preference of \(b_2\): \(a_2 \succ b_2 \succ a_3 \succ b_2 \succ a_1\)
preference of \(b_3\): \(a_2 \succ b_3 \succ a_3 \succ b_3 \succ a_1\)
The Gale-Shapley Algorithm (Example Run 3)

preference of $a_1$: $b_1 \succsim_{a_1} b_2 \succsim_{a_1} b_3$
preference of $a_2$: $b_1 \succsim_{a_2} b_2 \succsim_{a_2} b_3$
preference of $a_3$: $b_2 \succsim_{a_3} b_1 \succsim_{a_3} b_3$

preference of $b_1$: $a_3 \succsim b_1 a_2 \succsim b_1 a_1$
preference of $b_2$: $a_2 \succsim b_2 a_3 \succsim b_2 a_1$
preference of $b_3$: $a_2 \succsim b_3 a_3 \succsim b_3 a_1$
The Gale-Shapley Algorithm (Example Run 3)

preference of $a_1$: $b_1 \succ a_1 \ b_2 \succ a_1 \ b_3$
preference of $a_2$: $b_1 \succ a_2 \ b_2 \succ a_2 \ b_3$
preference of $a_3$: $b_2 \succ a_3 \ b_1 \succ a_3 \ b_3$

preference of $b_1$: $a_3 \succ b_1 \ a_2 \succ b_1 \ a_1$
preference of $b_2$: $a_2 \succ b_2 \ a_3 \succ b_2 \ a_1$
preference of $b_3$: $a_2 \succ b_3 \ a_3 \succ b_3 \ a_1$
The Gale-Shapley Algorithm (Example Run 3)

preference of $a_1$:  $b_1 \succ_{a_1} b_2 \succ_{a_1} b_3$
preference of $a_2$:  $b_1 \succ_{a_2} b_2 \succ_{a_2} b_3$
preference of $a_3$:  $b_2 \succ_{a_3} b_1 \succ_{a_3} b_3$

preference of $b_1$:  $a_3 \succ_{b_1} a_2 \succ_{b_1} a_1$
preference of $b_2$:  $a_2 \succ_{b_2} a_3 \succ_{b_2} a_1$
preference of $b_3$:  $a_2 \succ_{b_3} a_3 \succ_{b_3} a_1$
The Gale-Shapley Algorithm (Example Run 3)

preference of $a_1$: $b_1 \succsim_{a_1} b_2 \succsim_{a_1} b_3$
preference of $a_2$: $b_1 \succsim_{a_2} b_2 \succsim_{a_2} b_3$
preference of $a_3$: $b_2 \succsim_{a_3} b_1 \succsim_{a_3} b_3$

preference of $b_1$: $a_3 \succsim_{b_1} a_2 \succsim_{b_1} a_1$
preference of $b_2$: $a_2 \succsim_{b_2} a_3 \succsim_{b_2} a_1$
preference of $b_3$: $a_2 \succsim_{b_3} a_3 \succsim_{b_3} a_1$
The Gale-Shapley Algorithm (Example Run 3)

preference of $a_1$: $b_1 \succ a_1 b_2 \succ a_1 b_3$
preference of $a_2$: $b_1 \succ a_2 b_2 \succ a_2 b_3$
preference of $a_3$: $b_2 \succ a_3 b_1 \succ a_3 b_3$

preference of $b_1$: $a_3 \succ b_1 a_2 \succ b_1 a_1$
preference of $b_2$: $a_2 \succ b_2 a_3 \succ b_2 a_1$
preference of $b_3$: $a_2 \succ b_3 a_3 \succ b_3 a_1$
The Gale-Shapley Algorithm (Example Run 3)

preference of $a_1$: $b_1 \succ_{a_1} b_2 \succ_{a_1} b_3$
preference of $a_2$: $b_1 \succ_{a_2} b_2 \succ_{a_2} b_3$
preference of $a_3$: $b_2 \succ_{a_3} b_1 \succ_{a_3} b_3$
preference of $b_1$: $a_3 \succ_{b_1} a_2 \succ_{b_1} a_1$
preference of $b_2$: $a_2 \succ_{b_2} a_3 \succ_{b_2} a_1$
preference of $b_3$: $a_2 \succ_{b_3} a_3 \succ_{b_3} a_1$
The Gale-Shapley Algorithm (Example Run 3)

preference of $a_1$: $b_1 \succ a_1 \ b_2 \succ a_1 \ b_3$
preference of $a_2$: $b_1 \succ a_2 \ b_2 \succ a_2 \ b_3$
preference of $a_3$: $b_2 \succ a_3 \ b_1 \succ a_3 \ b_3$

preference of $b_1$: $a_3 \succ b_1 \ a_2 \succ b_1 \ a_1$
preference of $b_2$: $a_2 \succ b_2 \ a_3 \succ b_2 \ a_1$
preference of $b_3$: $a_2 \succ b_3 \ a_3 \succ b_3 \ a_1$
The Gale-Shapley Algorithm (Example Run 3)

preference of $a_1$: $b_1 \succ_a b_2 \succ_a b_3$
preference of $a_2$: $b_1 \succ_a b_2 \succ_a b_3$
preference of $a_3$: $b_2 \succ_a b_1 \succ_a b_3$

preference of $b_1$: $a_3 \succ_b a_2 \succ_b a_1$
preference of $b_2$: $a_2 \succ_b a_3 \succ_b a_1$
preference of $b_3$: $a_2 \succ_b a_3 \succ_b a_1$
The Gale-Shapley Algorithm

- **Claim:** the Gale-Shapley algorithm always terminates in $\leq n^2$ rounds
  
  - In every round, one candidate proposes to one other candidate
  - The proposing agents only go down in their preference list
  - So, every proposal happens at most once
  - Thus, there are at most $n^2$ rounds
The Gale-Shapley Algorithm

- **Claim:** when the Gale-Shapley algorithm terminates, all agents (on both sides) are matched
  - Every agent on the proposing side is matched
    - Suppose there is some proposing agent $c$ that is not matched
    - Then some non-proposing agent $d$ is also not matched
    - At some point $c$ proposed to $d$
    - Then $d$ would remain matched to some agent throughout the process
    - **Contradiction!**
  - Every agent on the non-proposing side is matched
    - By counting: there are equally proposing and non-proposing agents, and matchings are one-to-one
Claim: the matching returned by the Gale-Shapley algorithm is stable

Consider a pair \((c, d)\) that is not matched, where \(c\) is on the proposing side.

We distinguish two cases:

(1) Either \(c\) proposed to \(d\) at some point,
    ▶ Then \(d\) prefers their current partner to \(c\)
    (since \(c\) and \(d\) are not matched anymore)

(2) Or \(c\) never proposed to \(d\)
    ▶ Then \(c\) prefers their current partner to \(d\)
    ▶ In both cases, \((c, d)\) is not a blocking pair
Outcomes of the Gale-Shapley Algorithm

preference of $a_1$: $b_1 \succ a_1 b_2 \succ a_1 b_3$
preference of $a_2$: $b_1 \succ a_2 b_2 \succ a_2 b_3$
preference of $a_3$: $b_2 \succ a_3 b_1 \succ a_3 b_3$
preference of $b_1$: $a_3 \succ b_1 a_2 \succ b_1 a_1$
preference of $b_2$: $a_2 \succ b_2 a_3 \succ b_2 a_1$
preference of $b_3$: $a_2 \succ b_3 a_3 \succ b_3 a_1
The Gale-Shapley Algorithm

- What are the properties of the stable matchings returned by the Gale-Shapley algorithm?
  - Does it find all stable matchings?
  - Do the matchings that it finds satisfy certain fairness properties?
  - Can the agents manipulate the outcome by strategically reporting insincere preferences?
A- and B-Optimality

- A stable matching is **A-optimal** if every agent \( a \in A \) is matched to their most preferred agent among the agents \( b \) that they are matched with in any stable matching.

- A stable matching is **B-optimal** if every agent \( b \in B \) is matched to their most preferred agent among the agents \( a \) that they are matched with in any stable matching.

**Theorem (Gale, Shapley, 1962)**


A- and B-Optimality

**Theorem (Gale, Shapley, 1962)**


- **Idea:**
  - Suppose some \(a\) is matched to \(b\), but there is another stable matching \(\mu'\) where \(a\) is matched to \(b'\), and \(b' \succ_a b\)
  - So \(a\) proposed to \(b'\) before, but \(b'\) rejected
  - Assume that this was the first rejection of a “stable partner”
  - Let \(a'\) be the agent that \(b'\) chose over \(a\)
  - Then \(a'\) prefers \(b'\) over all “stable partners” (because \(b'\) rejecting \(a\) was the first rejection of a “stable partner”)
  - But then \(\mu'\) is not stable, as \((a', b')\) is a blocking pair
  - Contradiction!
Examples of Stable Matchings

- preference of $a_1$: $b_1 \succ a_1 \ b_2 \succ a_1 \ b_3$
- preference of $a_2$: $b_2 \succ a_2 \ b_3 \succ a_2 \ b_1$
- preference of $a_3$: $b_3 \succ a_3 \ b_1 \succ a_3 \ b_2$

- preference of $b_1$: $a_2 \succ b_1 \ a_3 \succ b_1 \ a_1$
- preference of $b_2$: $a_3 \succ b_2 \ a_1 \succ b_2 \ a_2$
- preference of $b_3$: $a_1 \succ b_3 \ a_2 \succ b_3 \ a_3$

- So the Gale-Shapley algorithm does not find all stable matchings
Strategizing for a mechanism $M$ consists of an agent $c$ reporting an insincere preference order $\succ'_c$ so that the outcome of $M$ for $\succ'_c$ is preferred by $c$ over the outcome of $M$ for $\succ_c$.

Is the Gale-Shapley algorithm strategyproof?

Theorem (Roth, 1982)

The A-proposing Gale-Shapley algorithm is strategyproof for A.
The B-proposing Gale-Shapley algorithm is strategyproof for B.

Strategizing

- Is there a stable matching mechanism that is strategyproof for both $A$ and $B$?

Theorem (Roth, 1982)

There exists no matching mechanism that is stable and strategyproof for both $A$ and $B$. 
Strategizing

Theorem (Roth, 1982)

There exists no matching mechanism that is stable and strategyproof for both A and B.

<table>
<thead>
<tr>
<th>Preference of $a_1$:</th>
<th>Preference of $b_1$:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_2 \succ a_1$</td>
<td>$a_1 \succ b_1$</td>
</tr>
<tr>
<td>$b_1 \succ a_1$</td>
<td>$b_1 \succ a_1$</td>
</tr>
<tr>
<td>$b_3 \succ a_1$</td>
<td>$b_3 \succ a_1$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Preference of $a_2$:</th>
<th>Preference of $b_2$:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_1 \succ a_2$</td>
<td>$a_3 \succ b_2$</td>
</tr>
<tr>
<td>$b_2 \succ a_2$</td>
<td>$a_1 \succ b_2$</td>
</tr>
<tr>
<td>$b_3 \succ a_2$</td>
<td>$a_2 \succ b_2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Preference of $a_3$:</th>
<th>Preference of $b_3$:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_1 \succ a_3$</td>
<td>$a_1 \succ b_3$</td>
</tr>
<tr>
<td>$b_2 \succ a_3$</td>
<td>$a_3 \succ b_3$</td>
</tr>
<tr>
<td>$b_3 \succ a_3$</td>
<td>$a_2 \succ b_3$</td>
</tr>
</tbody>
</table>
Strategizing

**Theorem (Roth, 1982)**

There exists no matching mechanism that is stable and strategyproof for both A and B.

- Preference of $a_1$: $b_2 \succ_a b_1 b_1 \succ_a a_1 b_3$
- Preference of $a_2$: $b_1 \succ_a a_2 b_2 \succ_a a_2 b_3$
- Preference of $a_3$: $b_1 \succ_a a_3 b_2 \succ_a a_3 b_3$
- Preference of $b_1$: $a_1 \succ_b b_1 a_3 \succ_b b_1 a_2$
- Preference of $b_2$: $a_3 \succ_b b_2 a_1 \succ_b b_2 a_2$
- Preference of $b_3$: $a_1 \succ_b b_3 a_2 \succ_b b_3 a_3$
Theorem (Roth, 1982)

There exists no matching mechanism that is stable and strategyproof for both A and B.

preference of $a_1$: $b_2 \succ a_1$ $b_1 \succ a_1$ $b_3$
preference of $a_2$: $b_1 \succ a_2$ $b_2 \succ a_2$ $b_3$
preference of $a_3$: $b_1 \succ a_3$ $b_2 \succ a_3$ $b_3$
preference of $b_1$: $a_1 \succ b_1$ $a_2 \succ b_1$ $a_3$
preference of $b_2$: $a_3 \succ b_2$ $a_1 \succ b_2$ $a_2$
preference of $b_3$: $a_1 \succ b_3$ $a_2 \succ b_3$ $a_3$
Theorem (Roth, 1982)

There exists no matching mechanism that is stable and strategyproof for both A and B.

preference of $a_1$: $b_2 \succ_a b_1 \succ_a b_3$

preference of $a_2$: $b_1 \succ_a b_2 \succ_a b_3$

preference of $a_3$: $b_1 \succ_a b_2 \succ_a b_3$

preference of $b_1$: $a_1 \succ_b a_3 \succ_b a_2$

preference of $b_2$: $a_3 \succ_b a_1 \succ_b a_2$

preference of $b_3$: $a_1 \succ_b a_2 \succ_b a_3$
Strategizing

Theorem (Roth, 1982)

There exists no matching mechanism that is stable and strategyproof for both A and B.

preference of $a_1$: $b_2 \succ a_1$ $b_3 \succ a_1$ $b_1$

preference of $a_2$: $b_1 \succ a_2$ $b_2 \succ a_2$ $b_3$

preference of $a_3$: $b_1 \succ a_3$ $b_2 \succ a_3$ $b_3$

preference of $b_1$: $a_1 \succ b_1$ $a_3 \succ b_1$ $a_2$

preference of $b_2$: $a_3 \succ b_2$ $a_1 \succ b_2$ $a_2$

preference of $b_3$: $a_1 \succ b_3$ $a_2 \succ b_3$ $a_3$
Stable Matching with Incomplete Lists

- Instead of specifying a full linear order over all agents on the other side, agents can specify a set of agents that they find acceptable, and give a linear preference order over them.

- An agent prefers remaining unmatched over being matched with an agent they find unacceptable.

- A matching is stable in this setting if there is no:
  - blocking pair, and
  - no agent that prefers being unmatched over their current match.

- The Gale-Shapley algorithm can be extended to the case with incomplete lists.
Stable Matching with Incomplete Lists and Ties

- We can also, in addition, allow agents to specify weak preference orders
  - i.e., allowing ties in their preferences

- In this case, stable matchings can have different size:

  \[
  \begin{align*}
  a_1 & \quad \bullet \quad b_1 \\
  a_2 & \quad \bullet \quad b_2 \\
  \end{align*}
  \]

  \[
  \begin{align*}
  a_1 & \quad \bullet \quad b_1 \\
  a_2 & \quad \bullet \quad b_2 \\
  \end{align*}
  \]

  preference of \( a_1 \): \( b_1 \)
  preference of \( a_2 \): \( b_1 \succ a_2 b_2 \)
  preference of \( b_1 \): \( a_1 \sim_{b_1} a_2 \)
  preference of \( b_2 \): \( a_2 \)
Stable Matching with Incomplete Lists and Ties

Max-SMTI

Input: two sets $A$ and $B$ of agents, for each agent $c \in A \cup B$ their preferences (with unacceptable agents and ties) over the agents in the other set, and a partial matching $\mu$.

Output: Is there a maximum size stable matching that includes $\mu$?

Theorem (Manlove et al., 2002)

Max-SMTI is NP-complete.

Stable Roommates

- Instead of a bipartite matching scenario with two sets $A$ and $B$, we have one single set $A$ of agents.

- Each agent $a$ specifies a preference over the other agents $A \setminus \{a\}$.

- In this setting, a stable matching does not always exist.

**Theorem (Irving, 1985)**

*There exists a polynomial-time algorithm to find a stable matching for the stable roommates problem, if it exists.*

Hospital-Residents Matching

- In this variant, $A$ is a set of residents and $B$ is a set of hospitals
  - Each hospital $b \in B$ has a capacity $c_b \in \mathbb{N}$ in addition to a preference over $A$
  - Both residents and hospitals can specify acceptable matches
- A matching then matches each resident $a$ to $\leq 1$ hospital, and each hospital $b$ to $\leq c_b$ residents
- An unmatched pair $(a, b)$ is blocking if $a$ prefers $b$ to their current match (or their currently being unmatched), and if $b$ prefers to be matched with $a$ (i.e., either if $b$ has a free spot, or if $b$ prefers $a$ over one of their current matches)
- The Gale-Shapley algorithm can be extended for HR matching
Further Topics in Matching

- Optimizing for the overall satisfaction of the agents over all stable matchings
  - Average satisfaction
  - Minimizing (maximum) regret

- Different notions of quality for matchings
  - Pareto optimal matchings
  - Popular matchings
  - etc.
(Stable) matching can also be applied to assign students to schools with a limited number of spots.

In this setting, the “preference” of schools is often based on a priority ranking or a tie-breaking lottery.

The Amsterdam school board decided to change their mechanism in 2015, resulting in a lot of controversy and even a court case!

This story features:

- The Gale-Shapley (or Deferred Acceptance) algorithm
- Strategizing
- Fairness

Read about it at https://goo.gl/26915E
Recap

- Stable Matching
- The Gale-Shapley Algorithm
- Strategizing
- Variants of Matching Problems
Let $A = \{a_1, a_2, a_3\}$ and let $B = \{b_1, b_2, b_3\}$.

Find preferences for $a_1, a_2, a_3$ (linear orders over $B$) and preferences for $b_1, b_2, b_3$ (linear orders over $A$) such that there is only one stable matching.

Perform the Gale-Shapley algorithm, both with $A$ proposing and with $B$ proposing.