Matching Residents to Hospitals

- **Goal**: Given a set of preferences among hospitals and medical school students, design a self-reinforcing admissions process.
- **Unstable pair**: applicant $x$ and hospital $y$ are unstable if:
  - $x$ prefers $y$ to their assigned hospital.
  - $y$ prefers $x$ to one of its admitted students.
- **Stable assignment**: Assignment with no unstable pairs.
  - Natural and desirable condition.
  - Individual self-interest will prevent any applicant/hospital deal from being made.

Stable Matching Problem

- **Goal**: Given $n$ men and $n$ women, find a "suitable" matching.
  - Participants rate members of opposite sex.
  - Each man lists women in order of preference from best to worst.
  - Each woman lists men in order of preference from best to worst.

Ment's Preference Profile

<table>
<thead>
<tr>
<th>Favorite</th>
<th>Least Favorite</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>2nd</td>
</tr>
<tr>
<td>Xavier</td>
<td>Amy</td>
</tr>
<tr>
<td>Yuri</td>
<td>Brenda</td>
</tr>
<tr>
<td>Zoran</td>
<td>Claire</td>
</tr>
</tbody>
</table>

Women's Preference Profile

<table>
<thead>
<tr>
<th>Favorite</th>
<th>Least Favorite</th>
<th>Favorite</th>
<th>Least Favorite</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>2nd</td>
<td>1st</td>
<td>2nd</td>
</tr>
<tr>
<td>Claire</td>
<td>Amy</td>
<td>Yuri</td>
<td>Xavier</td>
</tr>
<tr>
<td>Brenda</td>
<td>Claire</td>
<td>Zoran</td>
<td>Yuri</td>
</tr>
</tbody>
</table>

Stable Matching Problem

- **Perfect matching**: everyone is matched monogamously.
  - Each man gets exactly one woman.
  - Each woman gets exactly one man.
- **Stability**: no incentive for some pair of participants to undermine assignment by joint action.
  - In matching $M$, an unmatched pair $m-w$ is unstable if man $m$ and woman $w$ prefer each other to current partners.
  - Unstable pair $m-w$ could each improve by eloping.
- **Stable matching**: perfect matching with no unstable pairs.
- **Stable matching problem**: Given the preference lists of $n$ men and $n$ women, find a stable matching if one exists.

Q. Is assignment X-C, Y-B, Z-A stable?

- A. No. Brenda and Xavier will hook up.
Q. Is assignment X-A, Y-B, Z-C stable?  
A. Yes.

Stable Matching Problem

- Favorite least favorite favorite least favorite favorite
- Xavier Amy Brenda Claire
- Yuri Amy Brenda Claire
- Zoran Amy Brenda Claire

Stable Roommate Problem

Q. Do stable matchings always exist?  
A. Not obvious a priori.

Propose-And-Reject Algorithm

- Propose-and-reject algorithm. [Gale-Shapley 1962]
- Intuitive method that guarantees to find a stable matching.
- Initialize each person to free.
- while [some man is free and hasn’t proposed to every woman] {  
  Choose such a man m:
  W = [second woman on m’s list to whom m has not yet proposed if m is free]
  assign m and W to be engaged
  else if (W prefers m to her fiancé M):
    assign m and W to be engaged, and M to be free
  else:
    M rejects M
}

Proof of Correctness: Termination

- Observation 1. Men propose to women in decreasing order of preference.
- Observation 2. Once a woman is matched, she never becomes unmatched; she only “trades up.”
- Claim. Algorithm terminates after at most \( n^2 \) iterations of while loop.
- There are only \( n^2 \) possible proposals.

Proof of Correctness: Perfection

- Claim. All men and women get matched.
- Proof. (by contradiction)
  - Suppose, for sake of contradiction, that Zoran is not matched upon termination of algorithm.
  - Then some woman, say Amy, is not matched upon termination.
  - By Observation 2 (only trading up, never becoming unmatched), Amy was never proposed to.
  - But, Zoran proposes to everyone, since he ends up unmatched.

Proof of Correctness: Stability

- Claim. No unstable pairs.
- Proof. (by contradiction)
  - Suppose A-Z is an unstable pair; each prefers each other to partner in Gale-Shapley matching \( S^* \).
  - Case 1: Z never proposed to A.  
    - Z prefers his GS partner to A.  
    - A-Z is stable.
  - Case 2: Z proposed to A.  
    - A rejected Z (right away or later)  
    - A prefers her GS partner to Z.  
    - A-Z is stable.
  - In either case A-Z is stable, a contradiction.
Summary

- Stable matching problem. Given $n$ men and $n$ women, and their preferences, find a stable matching if one exists.
- Gale-Shapley algorithm. Guarantees to find a stable matching for any problem instance.
- Q. How to implement GS algorithm efficiently?
- Q. If there are multiple stable matchings, which one does GS find?

Implementation for Stable Matching Algorithms

- Problem size
  - $N=2n^2$ words
  - $2n^2$ people each with a preference list of length $n$
- Brute force algorithm
  - Try all $n!$ possible matchings
  - Do any of them work?
- Gale-Shapley Algorithm
  - $n^2$ iterations, each costing constant time as follows:

Efficient Implementation

- Efficient implementation. We describe $O(n^2)$ time implementation.
- Representing men and women.
  - Assume men are named 1, ... $n$.
  - Assume women are named 1, ..., $n$.
- Engagements.
  - Maintain a list of free men, e.g., in a queue.
  - Maintain two arrays $wife[i]$, and $husband[w]$.
    - set entry to 0 if unmatched
    - if $m$ matched to $w$ then $wife[m]=w$ and $husband[w]=m$
- Men proposing.
  - For each man, maintain a list of women, ordered by preference.
  - Maintain an array $count[m]$ that counts the number of proposals made by man $m$.

Efficient Implementation

- Women rejecting/accepting.
  - Does woman $w$ prefer $m$ to man $m'$?
  - For each woman, create $inverse$ of preference list of men.
  - Constant time access for each query after $O(n)$ preprocessing.

Understanding the Solution

- Q. For a given problem instance, there may be several stable matchings. Do all executions of Gale-Shapley yield the same stable matching? If so, which one?

  

<table>
<thead>
<tr>
<th>5</th>
<th>20</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>Mary</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>John</td>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>

- An instance with two stable matchings.
  - A-X, B-Y, C-Z.
  - A-Y, B-X, C-Z.

Understanding the Solution

- Q. For a given problem instance, there may be several stable matchings. Do all executions of Gale-Shapley yield the same stable matching? If so, which one?
- Def. Man $m$ is a valid partner of woman $w$ if there exists some stable matching in which they are matched.
- Man-optimal assignment. Each man receives best valid partner (according to his preferences).
- Claim. All executions of GS yield a man-optimal assignment, which is a stable matching!
  - No reason to believe that man-optimal assignment is perfect, let alone stable.
  - Simultaneously best for each and every man.
Woman Pessimal Assignment. Each woman receives worst valid partner.

Claim. GS finds woman-pessimal stable matching $S^*$.  

Proof. Suppose $A-Z$ matched in $S^*$, but $Z$ is not worst valid partner for $A$.  

There exists stable matching $S$ in which $A$ is paired with a man, say $Y$, whom she prefers to $Z$.  

Let $B$ be $Z$'s partner in $S$.  

In building $S^*$, when $Y$ is rejected by $A$, thus $Z$ prefers $A$ to $B$.  

Thus $A-Z$ is unstable in $S$.  

Stable Matching Summary

- Stable matching problem. Given preference profiles of $n$ men and $n$ women, find a stable matching.
- Gale-Shapley algorithm. Finds a stable matching in $O(n^2)$ time.
- Man-optimality. In version of GS where men propose, each man receives best valid partner.
- $Q$. Does man-optimality come at the expense of the women?

Application: Matching Residents to Hospitals

NRMP. (National Resident Matching Program)
- Original use just after WWII.
- Ides of March, 23,000+ residents.

Rural hospital dilemma.
- Certain hospitals (mainly in rural areas) were unpopular and declared unacceptable by many residents.
- Rural hospitals were under-subscribed in NRMP matching.
- How can we find stable matching that benefits “rural hospitals”?

Rural Hospital Theorem. Rural hospitals get exactly same residents in every stable matching!

Lessons Learned

- Powerful ideas learned in course.
  - Isolate underlying structure of problem.
  - Create useful and efficient algorithms.

- Potentially deep social ramifications.