Optimality Criteria for Matching with One-Sided Preferences

Richard Matthew McCutchen
Advisor: Samir Khuller
University of Maryland
Problem

• Given an *instance*:
  - Set of people
  - Set of positions available to them
  - Each person’s preference ordering of the positions
    • (Positions don’t have preferences; that would be two-way)

• Compute the “best” matching of people to positions

• Applications
  - TAs to classes
  - Netflix customers to their next DVDs
Approach

- Different matchings inevitably favor different people $\Rightarrow$ no obvious “best” matching
- Need an *optimality criterion*
  - An “optimal” matching should exist for every instance
  - Should be “fair”
  - Should be resistant to manipulation by people
  - Should admit an efficient algorithm to compute an optimal matching
Goal

• A computer program to solve real-world matching problems according to a good optimality criterion!

• Advantages
  - Fast/easy
  - Objective
  - Makes no mistakes
Example

<table>
<thead>
<tr>
<th></th>
<th>Cooking</th>
<th>Laundry</th>
<th>Dishes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Bob</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Carol</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

- Three people, three positions
- Numbers indicate preference ranks
Example

<table>
<thead>
<tr>
<th></th>
<th>Co</th>
<th>La</th>
<th>Di</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Bob</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Carol</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Co</th>
<th>La</th>
<th>Di</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Bob</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Carol</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

• Which is better?
Example

- Compare by majority vote
- Right matching is "popular"

<table>
<thead>
<tr>
<th></th>
<th>Co</th>
<th>La</th>
<th>Di</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Bob</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Carol</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Co</th>
<th>La</th>
<th>Di</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Bob</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Carol</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
Why voting?

• +1 or −1; ignores the distance between two positions on a preference list
  − Arguably less fair
  − Seems to be accepted for elections for public office

• Using difference of numerical ranks opens door to easy manipulation
  − Person can pad preference list with positions he/she won’t get to make algorithm pity him/her
  − Students once exploited MIT housing algorithm this way

• Until we have a safer way to consider distance, stick with voting
Finding a popular matching  
(Abraham, Irving, Kavitha, Mehlhorn; SODA 2005)

• A person’s *backup* position: her favorite position that isn’t anyone’s first choice

• Theorem: A matching is popular iff:
  − Every position that is someone’s first choice is filled, *and*
  − Each person gets either her *first choice* or her *backup*

<table>
<thead>
<tr>
<th>Example:</th>
<th>Cooking</th>
<th>Laundry</th>
<th>Dishes</th>
<th>Lawn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>2</td>
<td>3*</td>
<td>4</td>
</tr>
<tr>
<td>Bob</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>2*</td>
</tr>
<tr>
<td>Carol</td>
<td>3</td>
<td>1</td>
<td>2*</td>
<td>4</td>
</tr>
</tbody>
</table>
Finding a popular matching
(Abraham, Irving, Kavitha, Mehlhorn; SODA 2005)

- Max-match in graph of first choices and backups, then promote people into any unfilled first choices
Finding a popular matching
(Abraham, Irving, Kavitha, Mehlhorn; SODA 2005)

- Max-match in graph of first choices and backups, then promote people into any unfilled first choices

<table>
<thead>
<tr>
<th></th>
<th>Co</th>
<th>Ld</th>
<th>Di</th>
<th>Lw</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>2</td>
<td>3*</td>
<td>4</td>
</tr>
<tr>
<td>Bob</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>2*</td>
</tr>
<tr>
<td>Carol</td>
<td>3</td>
<td>1</td>
<td>2*</td>
<td>4</td>
</tr>
</tbody>
</table>

- More complicated algorithm works when preference orderings contain ties
No popular matching exists!

<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>Co</th>
<th>La</th>
<th>Di</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Carol</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Y</th>
<th>Co</th>
<th>La</th>
<th>Di</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Carol</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Z</th>
<th>Co</th>
<th>La</th>
<th>Di</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Carol</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
Unpopularity factor

• Helps us choose decent matchings rather than terrible ones when no popular matching exists
Unpopularity factor

• Helps us choose decent matchings rather than terrible ones when no popular matching exists

• \( N \) dominates \( M \) by a factor of \( u/v \), where:
  - \( u \) is \# people better off in \( N \)
  - \( v \) is \# people better off in \( M \)
Unpopularity factor

• Helps us choose decent matchings rather than terrible ones when no popular matching exists

• *N dominates M* by a factor of $u/v$, where:
  - $u$ is # people better off in $N$
  - $v$ is # people better off in $M$

• *Unpopularity factor* of $M$: Largest factor by which $M$ is dominated by any other matching
Unpopularity factor

• Helps us choose decent matchings rather than terrible ones when no popular matching exists

• $N$ dominates $M$ by a factor of $u/v$, where:
  - $u$ is # people better off in $N$
  - $v$ is # people better off in $M$

• *Unpopularity factor* of $M$: Largest factor by which $M$ is dominated by any other matching

• “Best” matching: least unpopularity factor

• Unpopularity factor $\leq 1 \Leftrightarrow$ popular
Example of U.F.

<table>
<thead>
<tr>
<th></th>
<th>Cooking</th>
<th>Laundry</th>
<th>Dishes</th>
<th>Cleaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Bob</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Carol</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Dave</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

- No popular matching exists
Example of U.F.

<table>
<thead>
<tr>
<th></th>
<th>Co</th>
<th>La</th>
<th>Di</th>
<th>Cl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Bob</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Carol</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Dave</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Co</th>
<th>La</th>
<th>Di</th>
<th>Cl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Bob</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Carol</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Dave</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

- Unpopularity factor of $M_1 = 3$
Example of U.F.

- Unpopularity factor of $M_2 = 2$
- $M_2$ is better than $M_1$
- $M_2$ is in fact best
Results

• Easy to calculate unpopularity factor of a given matching

• NP-hard to find the “best” matching (least unpopularity factor)
  - Can still find it exhaustively for few people and positions
Pressures

- **Pressure** of a position = # of people who can become better off if its occupant leaves
- Highest pressure = unpopularity factor

\[
\begin{array}{c|cccc}
\text{M}_2 & \text{Co} & \text{La} & \text{Di} & \text{Cl} \\
\hline
\text{Alice} & 1 & 2 & 3 & 4 \\
\text{Bob} & 1 & 2 & 3 & 4 \\
\text{Carol} & 1 & 2 & 3 & 4 \\
\text{Dave} & 1 & 2 & 4 & 3 \\
\end{array}
\]

- Carol: Cooking → Dishes
- Bob: Cooking → Dishes
- Dave: Cleaning → Dishes
- Carol: Cleaning → Dishes
- Dave: Cleaning → Dishes

Cycles: Cooking → Dishes → Cleaning → Dishes → Cooking
Finding U.F. of a matching

- Bellman-Ford shortest path algorithm
- Pressure edge: “length” = -1
- “Shortest” path length to a position gives its pressure
- Remember, highest pressure = unpopularity factor

Diagram:

- Cooking → Dishes
- Laundry → Cleaning
- Laundry → Dishes
- Cooking → Cleaning
- Laundry → Cooking
- Dishes → Cleaning
- Dishes → Laundry
- Cooking → Laundry
Finding matching of minimum U.F.

- Reduce 3SAT to the problem of finding the matching of minimum U.F. ⇒ it is NP-hard
- 3SAT solution ↔ matching of U.F. ≤ 2
- Gadgets confine pressures
- Analyze each gadget separately; a matching is acceptable iff it has pressure ≤ 2 in each
The reduction: Box

- To keep pressure $\leq 2$, can assign either wide or narrow(s) (but not one of each) inside box.

```
  n1  w  n2
```

```

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>z</th>
<th>u</th>
<th>l_w</th>
<th>l_{n1}</th>
<th>l_{n2}</th>
</tr>
</thead>
<tbody>
<tr>
<td>i_1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>i_2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>i_3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>w</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>n_1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>n_2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
</tbody>
</table>
```
The reduction: Variables

- Variable $\mapsto$ double-sided chain of boxes

- Box constraint gives us two options:
  - “True”: Assign “$x$” people inside boxes and “$\sim x$” people to linking positions
  - “False”: vice versa

- Leaves linking positions for satisfied variable references open
The reduction: Variables

- Variable $\mapsto$ double-sided chain of boxes

- Box constraint gives us two options:
  - "True": Assign "x" people inside boxes and "~x" people to linking positions
  - "False": vice versa

- Leaves linking positions for satisfied variable references open
The reduction: Variables

- Variable $\mapsto$ double-sided chain of boxes
  
  ![Diagram of double-sided chain of boxes]

- Box constraint gives us two options:
  - “True”: Assign “$x$” people inside boxes and “$\sim x$” people to linking positions
  - “False”: vice versa

- Leaves linking positions for satisfied variable references open
The reduction: Pool

- To keep pressure \( \leq 2 \), can assign at most two of the three linking people inside pool

![Diagram with three circles and arrows connecting them]

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>z</th>
<th>( l_{f_1} )</th>
<th>( l_{f_2} )</th>
<th>( l_{f_3} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_1 )</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>( f_2 )</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>–</td>
<td>1</td>
<td>–</td>
</tr>
<tr>
<td>( f_3 )</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>–</td>
<td>–</td>
<td>1</td>
</tr>
</tbody>
</table>
The reduction: Putting it together

- Clause $\rightarrow$ pool
  - Identify linking positions with those of box chains according to variable references
- Example: $a$ or $b$ or $a$; (not $b$) or (not $a$) or (not $b$)
The reduction: Putting it together

- Clause $\mapsto$ pool
  - Identify linking positions with those of box chains according to variable references
- Example: $a \lor b \lor a; \neg b \lor \neg a \lor \neg b$
- Set $a = \text{true}$, $b = \text{false}$
The reduction: Putting it together

- **Clause \( \mapsto \) pool**
  - Identify linking positions with those of box chains according to variable references
- **Example:** \( a \lor b \lor a; (\neg b) \lor (\neg a) \lor (\neg b) \)
- **Set** \( a = \text{true, } b = \text{false; assign pool linking people} \)
What to do about this?

- Can’t find matching of least unpopularity factor ⇒ the criterion is not useful for choosing matchings in practice
  - Open question: Is there an approximation algorithm?
- So try a different criterion!
Unpopularity margin

- \( N \) dominates \( M \) by a margin of \( u - v \) (instead of a factor of \( u/v \)); minimize the margin

- Differences:
  - Factor is based on worst pressure, a local property; margin is based (\textit{roughly}) on the sum of all pressures, a global property
  - Originally liked factor criterion because it handles Pareto efficiency more nicely (positive/0 \( \rightarrow \) infinite)
  - Margin criterion is better because one really bad, unfixable pressure doesn’t deter it from optimizing the rest of the matching
Finding U.M. of a matching

- *Min-cost flow* models reassignment of unit-size people, resulting in $-1$ and $+1$ costs (votes).

<table>
<thead>
<tr>
<th></th>
<th>Co</th>
<th>La</th>
<th>Di</th>
<th>Cl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Bob</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Carol</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Dave</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

All edge capacities are unit. Colors give costs: 0, $-1$, $+1$. 
Finding U.M. of a matching

- Flow represents difference from $M_2$ to $N_2$
- Min. cost = $-1 \Rightarrow$ unpopularity margin = 1

\begin{align*}
  &\text{Cooking} & \text{Laundry} & \text{Dishes} & \text{Cleaning} \\
  \text{Alice} & 1 & 2 & 3 & 4 \\
  \text{Bob} & 1 & 2 & 3 & 4 \\
  \text{Carol} & 1 & 2 & 3 & 4 \\
  \text{Dave} & 1 & 2 & 4 & 3
\end{align*}

All edge capacities are unit. Colors give costs: 0, $-1$, +1. Fat edges are used.
Finding matching of minimum U.M.

- Work in progress; neither algorithm nor NP-hardness proof yet
- Gadget-based reduction from 3SAT harder because we must account for all the pressures, not just the largest
Acknowledgments

• Samir Khuller, advisor
• Bobby Bhattacharjee
• Brian Dean
• Blair HS classmates, especially Nancy Zheng
• Dr. Torrence
Questions? Comments?