Recap

• Voting
  - \( n \) voters, \( m \) alternatives
  - Each voter \( i \) expresses a ranked preference \( \succ_i \)
  - Voting rule \( f \)
    - Takes as input the collection of preferences \( \succ \)
    - Returns a single alternative

• A plethora of voting rule
  - Plurality, Borda count, STV, Kemeny, Copeland, maximin, ...

Incentives

• Can a voting rule incentivize voters to truthfully report their preferences?

• Strategyproofness

➢ A voting rule is strategyproof if a voter cannot submit a false preference and get her more preferred alternative elected, irrespective of the preferences of other voters.

➢ Formally, a voting rule $f$ is strategyproof if there is no preference profile $\succ$, voter $i$, and false preference $\succ_i'$ s.t.

$$f(\succ_{-i}, \succ_i') >_i f(\succ)$$
Strategyproofness

- None of the rules we saw are strategyproof!

- Example: Borda Count
  - In the true profile, \( b \) wins
  - Voter 3 can make \( a \) win by pushing \( b \) to the end
Borda’s Response to Critics

My scheme is intended only for honest men!

Random 18th century French dude
Strategyproofness

• Are there any strategyproof rules?
  ➢ Sure

• Dictatorial voting rule
  ➢ The winner is always the most preferred alternative of voter $i$

• Constant voting rule
  ➢ The winner is always the same

• Not satisfactory (for most cases)
Three Properties

• **Strategyproof**: Already defined. No voter has an incentive to misreport.

• **Onto**: Every alternative can win under some preference profile.

• **Nondictatorial**: There is no voter $i$ such that $f(\sim)$ is always the alternative most preferred by voter $i$. 
Gibbard-Satterthwaite

• **Theorem:** For \( m \geq 3 \), no deterministic social choice function can be strategyproof, onto, and nondictatorial simultaneously 😞

• **Proof:** We will prove this for \( n = 2 \) voters.
  - Step 1: Show that SP implies “strong monotonicity” [Assignment?]
  - **Strong Monotonicity (SM):** If \( f(\succ) = a \), and \( \succ' \) is such that \( \forall i \in N, x \in A: a >_i x \Rightarrow a >'_i x \), then \( f(\succ') = a \).
    - If \( a \) still defeats every alternative it defeated in every vote in \( \succ \), it should still win.
Gibbard-Satterthwaite

• **Theorem:** For $m \geq 3$, no deterministic social choice function can be strategyproof, onto, and nondictatorial simultaneously 😞

• **Proof:** We will prove this for $n = 2$ voters.

  ➢ **Step 2:** Show that SP+onto implies “Pareto optimality” [Assignment?]  

  ➢ **Pareto Optimality (PO):** If $a \succ_i b$ for all $i \in N$, then $f(\succ) \neq b$.
    
    o If there is a different alternative that *everyone* prefers, your choice is not Pareto optimal (PO).
• **Proof for n=2:** Consider problem instance $I(a, b)$

<table>
<thead>
<tr>
<th>$\succ_1$</th>
<th>$\succ_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>b</td>
<td>a</td>
</tr>
</tbody>
</table>

Say $f(\succ_1, \succ_2) = a$

Due to strong monotonicity:

- PO: $f(\succ_1, \succ_2') \in \{a, b\}$
- SP: $f(\succ_1, \succ_2') \neq b$

$f(\succ'') = a$
Gibbard-Satterthwaite

• Proof for n=2:
  ➢ If \( f \) outputs \( a \) on instance \( I(a, b) \), voter 1 can get \( a \) elected whenever she puts \( a \) first.
    o In other words, voter 1 becomes dictatorial for \( a \).
    o Denote this by \( D(1, a) \).
  ➢ If \( f \) outputs \( b \) on \( I(a, b) \)
    o Voter 2 becomes dictatorial for \( b \), i.e., we have \( D(2, b) \).

• For every \( I(a, b) \), we have \( D(1, a) \) or \( D(2, b) \).
Gibbard-Satterthwaite

• Proof for n=2:
  ➢ On some \( I(a^*, b^*) \), suppose \( D(1, a^*) \) holds.
  ➢ Then, we show that voter 1 is a dictator. That is, \( D(1, b) \) must hold for every other \( b \) as well.
  ➢ Take \( b \neq a^* \). Because \( |A| \geq 3 \), there exists \( c \in A\setminus\{a^*, b\} \).
  ➢ Consider \( I(b, c) \). We either have \( D(1, b) \) or \( D(2, c) \).
  ➢ But \( D(2, c) \) is incompatible with \( D(1, a^*) \)
    ➢ Who would win if voter 1 puts \( a^* \) first and voter 2 puts \( c \) first?
  ➢ Thus, we have \( D(1, b) \), as required.
  ➢ QED!
Circumventing G-S

• Restricted preferences (later in the course)
  ➢ Not allowing all possible preference profiles
  ➢ Example: single-peaked preferences
    o Alternatives are on a line (say 1D political spectrum)
    o Voters are also on the same line
    o Voters prefer alternatives that are closer to them

• Use of money (later in the course)
  ➢ Require payments from voters that depend on the preferences they submit
  ➢ Prevalent in auctions
Circumventing G-S

• Randomization (later in this lecture)

• Equilibrium analysis
  ➢ How will strategic voters act under a voting rule that is not strategyproof?
  ➢ Will they reach an “equilibrium” where each voter is happy with the (possibly false) preference she is submitting?

• Restricting information
  ➢ Can voters successfully manipulate if they don’t know the votes of the other voters?
Circumventing G-S

• Computational complexity
  ➢ So we need to use a rule that is manipulable.
  ➢ Can we make it NP-hard for voters to manipulate? [Bartholdi et al., SC&W 1989]
  ➢ NP-hardness can be a good thing!

• $f$-MANIPULATION problem (for a given voting rule $f$):
  ➢ Input: Manipulator $i$, alternative $p$, votes of other voters (non-manipulators)
  ➢ Output: Can the manipulator cast a vote that makes $p$ uniquely win under $f$?
Example: Borda

- Can voter 3 make $a$ win?

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>b</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>a</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td></td>
<td>d</td>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>b</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td></td>
<td>a</td>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td></td>
<td>d</td>
<td>d</td>
<td>b</td>
</tr>
</tbody>
</table>
A Greedy Algorithm

• **Goal:** The manipulator wants to make alternative $p$ win uniquely

• **Algorithm:**
  - Rank $p$ in the first place
  - While there are unranked alternatives:
    - If there is an alternative that can be placed in the next spot without preventing $p$ from winning, place this alternative.
    - Otherwise, return false.
### Example: Borda

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>c</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>c</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>c</td>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>d</td>
<td>d</td>
<td>b</td>
</tr>
</tbody>
</table>
Example: Copeland

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>b</td>
<td>e</td>
<td>e</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>a</td>
<td>c</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>c</td>
<td>d</td>
<td>b</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>d</td>
<td>e</td>
<td>a</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>e</td>
<td>c</td>
<td>d</td>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>

Preference profile

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>-</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>3</td>
<td>-</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>c</td>
<td>2</td>
<td>2</td>
<td>-</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>e</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>-</td>
</tr>
</tbody>
</table>

Pairwise elections
## Example: Copeland

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>e</td>
<td>e</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>a</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>d</td>
<td>b</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td>a</td>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>c</td>
<td>d</td>
<td>d</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Preference profile

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>-</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>3</td>
<td>-</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>c</td>
<td>2</td>
<td>3</td>
<td>-</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>e</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>-</td>
</tr>
</tbody>
</table>

### Pairwise elections
Example: Copeland

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>e</td>
<td>e</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>a</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>c</td>
<td>d</td>
<td>b</td>
<td>b</td>
<td>d</td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td>a</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>c</td>
<td>d</td>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>

Preference profile

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>-</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>b</td>
<td>3</td>
<td>-</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>c</td>
<td>2</td>
<td>3</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>e</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Pairwise elections
Example: Copeland

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>e</td>
<td>e</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>a</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>d</td>
<td>b</td>
<td>b</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td>a</td>
<td>a</td>
<td>e</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>c</td>
<td>d</td>
<td>d</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Preference profile

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>-</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>3</td>
<td>-</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>c</td>
<td>2</td>
<td>3</td>
<td>-</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>e</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>-</td>
</tr>
</tbody>
</table>

Pairwise elections
### Example: Copeland

#### Preference profile

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>e</td>
<td>e</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>a</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>d</td>
<td>b</td>
<td>b</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td>a</td>
<td>a</td>
<td>e</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>c</td>
<td>d</td>
<td>d</td>
<td>b</td>
<td></td>
</tr>
</tbody>
</table>

#### Pairwise elections

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>-</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>3</td>
<td>-</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>c</td>
<td>2</td>
<td>3</td>
<td>-</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>e</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>-</td>
</tr>
</tbody>
</table>
When does this work?

• **Theorem** [Bartholdi et al., SCW 89]:

  Fix voter $i$ and votes of other voters. Let $f$ be a rule for which $\exists$ function $s(\succ_i, x)$ such that:

  1. For every $\succ_i$, $f$ chooses a candidate $x$ that **uniquely** maximizes $s(\succ_i, x)$.

  2. $\{y : x \succ_i y\} \subseteq \{y : x \succ_i' y\} \Rightarrow s(\succ_i, x) \leq s(\succ_i', x)$

  Then the greedy algorithm solves $f$-MANIPULATION correctly.

• **Question:** What is the function $s$ for plurality?
Proof of the Theorem

• Say the algorithm creates a partial ranking $\succ_i$ and then fails, i.e., every next choice prevents $p$ from winning.

• Suppose for contradiction that $\succ'_i$ could make $p$ uniquely win.

• $U \leftarrow$ alternatives not ranked in $\succ_i$

• $u \leftarrow$ highest ranked alternative in $U$ according to $\succ'_i$

• Complete $\succ_i$ by adding $u$ next, and then other alternatives arbitrarily.

\[ U = \{a, c\} \]
Proof of the Theorem

• $s(>_{i}, p) \geq s(>_{i}', p)$
  ➢ Property 2

• $s(>_{i}', p) > s(>_{i}', u)$
  ➢ Property 1 & $p$ wins under $>_{i}'$

• $s(>_{i}', u) \geq s(>_{i}, u)$
  ➢ Property 2

• Conclusion
  ➢ Putting $u$ in the next position wouldn’t have prevented $p$ from winning
  ➢ So the algorithm should have continued
Hard-to-Manipulate Rules

• Natural rules
  ➢ Copeland with second-order tie breaking
    [Bartholdi et al. SCW 89]
    o In case of a tie, choose the alternative for which the sum of
      Copeland scores of defeated alternatives is the largest
  ➢ STV [Bartholdi & Orlin, SCW 91]
  ➢ Ranked Pairs [Xia et al., IJCAI 09]
    o Iteratively lock in pairwise comparisons by their margin of victory
      (largest first), ignoring any comparison that would form cycles.
    o Winner is the top ranked candidate in the final order.

• Can also “tweak” easy to manipulate voting rules
  [Conitzer & Sandholm, IJCAI 03]
Example: Ranked Pairs
Example: Ranked Pairs
Example: Ranked Pairs
Example: Ranked Pairs
Example: Ranked Pairs
Example: Ranked Pairs

```
Example: Ranked Pairs

a -> b
b -> c
c -> d
d -> a
```

2

34
Example: Ranked Pairs

a → b → d → c
Randomized Voting Rules

• Take as input a preference profile, output a distribution over alternatives

• To think about successful manipulations, we need numerical utilities

• \( \succ_i \) is consistent with \( u_i \) if
  \[
  a \succ_i b \iff u_i(a) > u_i(b)
  \]

• Strategyproofness: For all \( i, u_i, \succ_{-i}, \) and \( \succ_i' \)
  \[
  \mathbb{E} \left[ u_i \left( f \left( \succ' \right) \right) \right] \geq \mathbb{E} \left[ u_i \left( f \left( \succ_{-i}, \succ_i' \right) \right) \right]
  \]
  where \( \succ_i \) is consistent with \( u_i \).
Randomized Voting Rules

• A (deterministic) voting rule is
  ➢ unilateral if it only depends on one voter
  ➢ duple if its range contains at most two alternatives

• A probability mixture $f$ over rules $f_1, \ldots, f_k$ is a rule given by some probability distribution $(\alpha_1, \ldots, \alpha_k)$ s.t. on every profile $\succ$, $f$ returns $f_j(\succ)$ w.p. $\alpha_j$. 
Randomized Voting Rules

• Theorem [Gibbard 77]:
  A randomized voting rule is strategyproof only if it is a probability mixture over unilaterals and duples.

• Example:
  ➢ With probability 0.5, output the top alternative of a randomly chosen voter
  ➢ With the remaining probability 0.5, output the winner of the pairwise election between $a^*$ and $b^*$

• Question: What is a probability mixture over unilaterals and duples that is not strategyproof?
Approximating Voting Rules

- **Idea:** Can we use strategyproof voting rules to approximate popular voting rules?

- Fix a rule (e.g., Borda) with a clear notion of score denoted $sc(\succ, a)$

- A randomized voting rule $f$ is a $c$-approximation to $sc$ if for every profile $\succ$
  \[
  \frac{\mathbb{E}[sc(\succ, f(\succ))]}{\max_a sc(\succ, a)} \geq c
  \]
Approximating Borda

• **Question:** How well does choosing a random alternative approximate Borda?
  1. $\Theta(1/n)$
  2. $\Theta(1/m)$
  3. $\Theta(1/\sqrt{m})$
  4. $\Theta(1)$

• **Theorem [Procaccia 10]:**
  No strategyproof voting rule gives $\frac{1}{2} + \omega\left(\frac{1}{\sqrt{m}}\right)$ approximation to Borda.
Interlude: Zero-Sum Games

![Game Matrix]

-1 1
1 -1
Interlude: Minimiax Strategies

• A minimax strategy for a player is
  ➢ a (possibly) randomized choice of action by the player
  ➢ that minimizes the expected loss (or maximizes the expected gain)
  ➢ in the worst case over the choice of action of the other player

• In the previous game, the minimax strategy for each player is \((1/2, 1/2)\). Why?
• In the game above, if the shooter uses \((p, 1 - p)\):
  ➢ If goalie jumps left: \(p \cdot \left(-\frac{1}{2}\right) + (1 - p) \cdot 1 = 1 - \frac{3}{2}p\)
  ➢ If goalie jumps right: \(p \cdot 1 + (1 - p) \cdot (-1) = 2p - 1\)
  ➢ Shooter chooses \(p\) to maximize \(\min \left\{ 1 - \frac{3p}{2}, 2p - 1 \right\}\)
Interlude: Minimax Theorem

• Theorem [von Neumann, 1928]:

Every 2-player zero-sum game has a unique value $v$ such that

- Player 1 can guarantee value at least $v$
- Player 2 can guarantee loss at most $v$
Yao’s Minimax Principle

• Rows as inputs
• Columns as deterministic algorithms
• Cell numbers = running times
• Best randomized algorithm

➢ Minimax strategy for the column player

\[
\min_{\text{rand algo}} \max_{\text{input}} E[time] = \max_{\text{dist over inputs}} \min_{\text{det algo}} E[time]
\]
Yao’s Minimax Principle

• To show a lower bound $T$ on the best worst-case running time achievable through randomized algorithms:
  ➢ Show a “bad” distribution over inputs $D$ such that every deterministic algorithm takes time at least $T$ on average, when inputs are drawn according to $D$

\[
\min_{\text{rand algo}} \max_{\text{input}} E[\text{time}] = \max_{\text{dist over inputs}} \min_{\text{det algo}} E[\text{time}]
\]
## Randomized Voting Rules

<table>
<thead>
<tr>
<th></th>
<th>( \preceq_1 )</th>
<th>( \ldots )</th>
<th>( \ldots )</th>
<th>( \ldots )</th>
<th>( \ldots )</th>
<th>( \preceq_t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( U_1 )</td>
<td>( \frac{1}{15} )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \frac{2}{21} )</td>
</tr>
<tr>
<td>( U_k )</td>
<td>( \frac{7}{15} )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \frac{5}{21} )</td>
</tr>
<tr>
<td>( D_1 )</td>
<td>( \frac{4}{15} )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \frac{8}{21} )</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( D_s )</td>
<td>( \frac{13}{15} )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \frac{17}{21} )</td>
</tr>
</tbody>
</table>

**Approximation ratio**

\[ \frac{5}{21} \]
Randomized Voting Rules

- Rows = unilaterals and duples
- Columns = preference profiles
- Cell numbers = approximation ratios

- The expected ratio of the best strategyproof rule (by Gibbard’s theorem, distribution over unilaterals and duples) is at most...
  - The expected ratio of the best unilateral or duple rule when profiles are drawn from a “bad” distribution $D$
A Bad Distribution

• \( m = n + 1 \)

• Choose a random alternative \( x^* \)

• Each voter \( i \) chooses a random number \( k_i \in \{1, \ldots, \sqrt{m}\} \) and places \( x^* \) in position \( k_i \)

• The other alternatives are ranked cyclically

\[
\begin{array}{ccc}
1 & 2 & 3 \\
\hline
\text{c} & \text{b} & \text{d} \\
\text{b} & \text{a} & \text{b} \\
\text{a} & \text{d} & \text{c} \\
\text{d} & \text{c} & \text{a} \\
\end{array}
\]

\[
x^* = b \\
k_1 = 2 \\
k_2 = 1 \\
k_3 = 2
\]
A Bad Distribution

• **Question:** What is the best lower bound on \( sc(\succ, x^*) \) that holds for every profile \( \succ \) generated under this distribution?

1. \( \sqrt{n} \)
2. \( \sqrt{m} \)
3. \( n \cdot (m - \sqrt{m}) \)
4. \( n \cdot m \)
A Bad Distribution

• How bad are other alternatives?
  ➢ For every other alternative $x$, $sc(\rightarrow, x) \sim \frac{n(m-1)}{2}$

• How surely can a unilateral/duple rule return $x^*$?
  ➢ Unilateral: By only looking at a single vote, the rule is essentially guessing $x^*$ among the first $\sqrt{m}$ positions, and captures it with probability at most $1/\sqrt{m}$.
  ➢ Duple: By fixing two alternatives, the rule captures $x^*$ with probability at most $2/m$.

• Putting everything together...
Quantitative GS Theorem

• Regarding the use of NP-hardness to circumvent GS
  ➢ NP-hardness is hardness in the worst case
  ➢ What happens in the average case?

• Theorem [Mossel-Racz ‘12]:
  For every voting rule that is at least $\epsilon$-far from being a dictatorship or having range of size 2, the probability that a profile chosen uniformly at random admits a manipulation is at least $p(n, m, \frac{1}{\epsilon})$ for some polynomial $p$. 
Coalitional Manipulations

• What if multiple voters collude to manipulate?
  ➢ The following result applies to a wide family of voting rules called “generalized scoring rules”.

• Theorem [Conitzer-Xia ‘08]:

  Powerful
  \[ \Theta(\sqrt{n}) \]

  Powerless

  Powerful = can manipulate with high probability
Interesting Tidbit

• Detecting a manipulable profile versus finding a beneficial manipulation

• **Theorem** [Hemaspaandra, Hemaspaandra, Menton ‘12]  
  If integer factoring is NP-hard, then there exists a generalized scoring rule for which:
  - We can efficiently check if there exists a beneficial manipulation.
  - But finding such a manipulation is NP-hard.
Next Lecture

• Frameworks to compare voting rules
  ➢ Even if we assume that voters will reveal their true preferences, we still don’t know if there is one “right” way to choose the winner.
  ➢ There are reasonable profiles where most prominent voting rules return different winners [Assignment?]