CSC2556
Lecture 2
Manipulation in Voting

Credit for many visuals: Ariel D. Procaccia
Recap

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

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

CSC2556 - Nisarg Shah
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.

$$f(\succ_{-i}, \succ'_{i}) \succ_{i} f(\succ)$$
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 $f(\succ)$ 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 \( f(\succeq) = a \), and \( \succeq' \) is such that \( \forall i \in N, x \in A: a \succ_i x \Rightarrow a \succ'_i x \), then \( f(\succeq') = a \).

    o If \( a \) still defeats every alternative it defeated in every vote in \( \succeq \), 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 $I(a, b)$

\[
\begin{array}{|c|c|}
\hline
>_{1} & >_{2} \\
\hline
a & b \\
b & a \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
>_{1} & >'_{2} \\
\hline
a & b \\
b & a \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
>_{1}'' & >_{2}'' \\
\hline
a & A \\
A & N \\
N & Y \\
\hline
\end{array}
\]

$f(>_{1}, >_{2}) \in \{a, b\}$

$\Rightarrow$ PO

Say $f(>_{1}, >_{2}) = a$

$f(>_{1}, >'_{2}) = a$

- PO: $f(>_{1}, >'_{2}) \in \{a, b\}$
- SP: $f(>_{1}, >'_{2}) \neq b$

- Due to strong monotonicity

\[
\begin{array}{|c|c|}
\hline
>_{1}'' & >_{2}'' \\
\hline
a & A \\
A & N \\
N & Y \\
\hline
\end{array}
\]

$f(>''') = a$
• **Proof for n=2:**
  - If $f$ outputs $a$ on instance $I(a, b)$, voter 1 can get $a$ elected whenever she puts $a$ first.
    - In other words, voter 1 becomes dictatorial for $a$.
    - Denote this by $D(1, a)$.
  - If $f$ outputs $b$ on $I(a, b)$
    - 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 $c \in A \{a^*, b\}$.
  ➢ Consider $I(b, c)$. We either have $D(1, b)$ or $D(2, c)$.
  ➢ But $D(2, c)$ is incompatible with $D(1, a^*)$
    o 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 the rule 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>b</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>d</td>
<td></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>b</td>
<td>b</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>c</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>d</td>
<td>b</td>
<td></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></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>b</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>c</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>4</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>1</td>
<td>b</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>3</td>
<td>c</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>4</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>1</td>
<td>b</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>3</td>
<td>c</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>4</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>1</td>
<td>b</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>3</td>
<td>c</td>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>4</td>
<td>d</td>
<td>d</td>
<td></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>b</td>
<td>e</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></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>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>2</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>0</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>e</td>
<td>a</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>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>0</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></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>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>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>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>3</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Pairwise elections
## Example: Copeland

### Preference profile

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

CSC2556 - Nisarg Shah
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.
Proof of the Theorem

• \( s(\succ_i, p) \geq s(\succ'_i, p) \)
  ➢ Property 2

• \( s(\succ'_i, p) > s(\succ'_i, u) \)
  ➢ Property 1 & \( p \) wins under \( \succ'_i \)

• \( s(\succ'_i, u) \geq s(\succ_i, u) \)
  ➢ 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

![Graph showing relationships between elements a, b, c, and d with ranked pairs.]

- a ranks higher than b by 6
- b ranks higher than c by 4
- c ranks higher than d by 2
- d ranks higher than a by 8
- a ranks higher than d by 10
- b ranks higher than a by 12
Example: Ranked Pairs
Example: Ranked Pairs

![Diagram](image-url)
Example: Ranked Pairs
Example: Ranked Pairs

a → b
a → d
d → a
d → c
c → d
b → a
b → c
b → d
4
2
Example: Ranked Pairs
Example: Ranked Pairs
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 $\succ'_i$
  \[ \mathbb{E} \left[u_i \left(f(\succ')\right)\right] \geq \mathbb{E} \left[u_i \left(f(\succ_{-i}, \succ'_i)\right)\right] \]
  where $\succ_i$ is consistent with $u_i$. 
Randomized Voting Rules

• A (deterministic) voting rule is
  ➢ **unilateral** if it only depends on one voter
  ➢ **dupe** 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

![Zero-Sum Game Diagram]
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?
Interlude: Minimiax Strategies

- 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 \(\min \left\{ 1 - \frac{3p}{2}, 2p - 1 \right\}\)
Theorem
[von Neumann, 1928]:

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

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

• Rows as inputs
• Columns as deterministic algorithms
• Cell numbers = running times
• Best randomized algorithm
  ➢ Minimax strategy for the column player

\[
\min_{\text{rand algo}} \max_{\text{input}} E[\text{time}] = \max_{\text{dist over inputs}} \min_{\text{det algo}} E[\text{time}]
\]
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 $D$

$$\min_{\text{rand algo}} \max_{\text{input}} E[\text{time}] = \max_{\text{dist over inputs}} \min_{\text{det algo}} E[\text{time}]$$
Next Lecture

• Finish the proof of Borda inapproximability result

• We’ll later see other ways to circumvent G-S.

• 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?]