Choix Social Computationnel

Nicolas Maudet
nicolas.maudet@lip6.fr

Université Pierre et Marie Curie

05 Septembre 2011
Motivation
Quest for the “best” voting system

Figure: Referendum on Alternative Vote (UK, 2011)
Motivation
Manipulating the system: Gerrymandering

Figure: The salamander of Elbridge Gerry (1812)
Motivation
Online voting systems

**Figure:** Choice of a restaurant (maybe open on Sunday, to check)
Motivation

Meta-search engines

Figure: Aggregating search results
Plan du cours

1. Voting Rules
2. Basics of Social Choice
3. Computing Winners
4. Manipulation
5. Communication Complexity
6. Voting with Incomplete Profiles
7. Other Topics
Voting

1. a finite set of voters $\mathcal{A} = \{1, \ldots, n\}$;
2. a finite set of candidates (alternatives) $\mathcal{X}$;
3. a profile = a preference relation (= linear order) on $\mathcal{X}$ for each voter
   \[ P = (V_1, \ldots, V_n) = (\succ_1, \ldots, \succ_n) \]
   \[ V_i \ (\text{or } \succ_i) = \text{vote expressed by voter } i. \]
4. $\mathcal{P}^n$ set of all profiles.
Voting

1. a finite set of voters \( \mathcal{A} = \{1, \ldots, n\} \);
2. a finite set of candidates (alternatives) \( \mathcal{X} \);
3. a profile = a preference relation (= linear order) on \( \mathcal{X} \) for each voter
   \[ P = (V_1, \ldots, V_n) = (\succ_1, \ldots, \succ_n) \]
   \( V_i \) (or \( \succ_i \)) = vote expressed by voter \( i \).
4. \( \mathcal{P}^n \) set of all profiles.

- **Voting rule** \( F : \mathcal{P}^n \rightarrow \mathcal{X} \)
  \( F(V_1, \ldots, V_n) = \) socially preferred (elected) candidate
- **Voting correspondence** \( C : \mathcal{P}^n \rightarrow 2^\mathcal{X} \setminus \{\emptyset\} \)
  \( C(V_1, \ldots, V_n) = \) set of socially preferred candidates.
- **Social welfare function** \( H : \mathcal{P}^n \rightarrow \mathcal{P} \)
  \( H(V_1, \ldots, V_n) = \) social preference relation (\( \succ_P \))

**Note**: Rules can be obtained from correspondences by tie-breaking (usually by using a predefined priority order on candidates).
Positional scoring rules

- \( n \) voters, \( p \) candidates
- fixed list of \( p \) integers \( s_1 \geq \ldots \geq s_p \)
- voter \( i \) ranks candidate \( x \) in position \( j \) \( \Rightarrow \) \( \text{score}_i(x) = s_j \)
- winner: candidate maximizing \( s(x) = \sum_{i=1}^{n} \text{score}_i(x) \)

Examples:

- \( s_1 = 1, s_2 = \ldots = s_m = 0 \Rightarrow \text{plurality} \);
- \( s_1 = s_2 = \ldots = s_{m-1} = 1, s_m = 0 \Rightarrow \text{veto} \);
- \( s_1 = m-1, s_2 = m-2, \ldots s_m = 0 \Rightarrow \text{Borda} \).

<table>
<thead>
<tr>
<th>2 voters</th>
<th>1 voter</th>
<th>1 voter</th>
<th>plurality</th>
<th>Borda</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>a</td>
<td>d</td>
<td>a ( \mapsto ) 1</td>
<td>a ( \mapsto ) 6</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>a</td>
<td>b ( \mapsto ) 0</td>
<td>b ( \mapsto ) 7</td>
</tr>
<tr>
<td>a</td>
<td>d</td>
<td>b</td>
<td>c ( \mapsto ) 2</td>
<td>c ( \mapsto ) 6</td>
</tr>
<tr>
<td>d</td>
<td>c</td>
<td>c</td>
<td>d ( \mapsto ) 1</td>
<td>d ( \mapsto ) 4</td>
</tr>
</tbody>
</table>

- c winner
- b winner
Condorcet winner

\[ N(x, y) = \{ i \mid x \succ_i y \} \] set of voters who prefer \( x \) to \( y \).

\#N(x, y) number of voters who prefer \( x \) to \( y \).

Condorcet winner

for \( P = \langle \succ_1, \ldots, \succ_n \rangle \): a candidate \( x \) such that \( \forall y \neq x, \#N(x, y) > \frac{n}{2} \)

(a candidate who beats any other candidate by a majority of votes).

\begin{align*}
& a \quad d \quad c \\
& b \quad b \quad a \\
& d \quad c \quad d
\end{align*}

Majority graph

2 voters out of 3 : \( a \succ b \)
2 voters out of 3 : \( c \succ a \)
2 voters out of 3 : \( a \succ d \)
2 voters out of 3 : \( b \succ c \)
2 voters out of 3 : \( b \succ d \)
2 voters out of 3 : \( d \succ c \)
Condorcet winner

\[ N(x, y) = \{ i \mid x \succ_i y \} \] set of voters who prefer \( x \) to \( y \).
\[ \#N(x, y) \] number of voters who prefer \( x \) to \( y \).

Condorcet winner

for \( P = \langle \succ_1, \ldots, \succ_n \rangle \): a candidate \( x \) such that \( \forall y \neq x, \#N(x, y) > \frac{n}{2} \) (a candidate who beats any other candidate by a majority of votes).

Majority graph

\[ \Rightarrow \text{No Condorcet winner} \]
Condorcet winner

\[ N(x, y) = \{ i \mid x \succ_i y \} \] set of voters who prefer \( x \) to \( y \).
\[ \#N(x, y) \] number of voters who prefer \( x \) to \( y \).

Condorcet winner

for \( P = \langle \succ_1, \ldots, \succ_n \rangle \): a candidate \( x \) such that \( \forall y \neq x, \#N(x, y) > \frac{n}{2} \)
(a candidate who beats any other candidate by a majority of votes).

\[
\begin{array}{ccc}
  a & d & c \\
  b & b & a \\
  d & c & d \\
\end{array}
\]

2 voters out of 3 : \( a \succ b \)
2 voters out of 3 : \( a \succ c \)
2 voters out of 3 : \( a \succ d \)
2 voters out of 3 : \( b \succ c \)
2 voters out of 3 : \( b \succ d \)
2 voters out of 3 : \( d \succ c \)

Majority graph
Condorcet winner

\[ N(x, y) = \{ i \mid x \succ_i y \} \] set of voters who prefer \( x \) to \( y \).
\[ \#N(x, y) \] number of voters who prefer \( x \) to \( y \).

Condorcet winner

for \( P = \langle \succ_1, \ldots, \succ_n \rangle \): a candidate \( x \) such that \( \forall y \neq x, \#N(x, y) > \frac{n}{2} \)
(a candidate who beats any other candidate by a majority of votes).

2 voters out of 3 : \( a \succ b \)
2 voters out of 3 : \( a \succ c \)
2 voters out of 3 : \( a \succ d \)
2 voters out of 3 : \( b \succ c \)
2 voters out of 3 : \( b \succ d \)
2 voters out of 3 : \( d \succ c \)

\( a \) is the Condorcet winner
Condorcet-consistent rules
The Copeland rule

- **Consistency with Condorcet**: the voting rule should elect the Condorcet winner whenever there is one.

- **Example: Copeland rule**
  get 1 pt for each pairwise win, $\frac{1}{2}$ for a tie, 0 otherwise

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

- 4 voters out of 5: $a \succ b$
- 3 voters out of 5: $c \succ a$
- 4 voters out of 5: $a \succ d$
- 3 voters out of 5: $b \succ c$
- 4 voters out of 5: $b \succ d$
- 3 voters out of 5: $d \succ c$

**Majority graph**

```
4 voters out of 5 : $a \succ b$
3 voters out of 5 : $c \succ a$
4 voters out of 5 : $a \succ d$
3 voters out of 5 : $b \succ c$
4 voters out of 5 : $b \succ d$
3 voters out of 5 : $d \succ c$
```
Condorcet-consistent rules
The Copeland rule

- **Consistency with Condorcet**: the voting rule should elect the Condorcet winner whenever there is one.

- **Example**: Copeland rule
  - get 1 pt for each pairwise win, $\frac{1}{2}$ for a tie, 0 otherwise

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

4 voters out of 5: $a \succ b$
3 voters out of 5: $c \succ a$
4 voters out of 5: $a \succ d$
3 voters out of 5: $b \succ c$
4 voters out of 5: $b \succ d$
3 voters out of 5: $d \succ c$

Majority graph:

- $C(a) = 2$
- $C(b) = 2$
- $C(c) = 1$
- $C(d) = 1$
Consistency with Condorcet: the voting rule should elect the Condorcet winner whenever there is one.

Example: Simpson rule
pick the candidate who minimizes the max pairwise defeat

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

4 voters out of 5:  \( a \succ b \)
3 voters out of 5:  \( c \succ a \)
4 voters out of 5:  \( a \succ d \)
3 voters out of 5:  \( b \succ c \)
4 voters out of 5:  \( b \succ d \)
3 voters out of 5:  \( d \succ c \)
Condorcet-consistent rules
The Simpson rule

▶ **Consistency with Condorcet**: the voting rule should elect the Condorcet winner whenever there is one.

▶ **Example**: *Simpson rule*
pick the candidate who minimizes the max pairwise defeat

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

4 voters out of 5: \( a \succ b \)
3 voters out of 5: \( c \succ a \)
4 voters out of 5: \( a \succ d \)
3 voters out of 5: \( b \succ c \)
4 voters out of 5: \( b \succ d \)
3 voters out of 5: \( d \succ c \)

(Wighted) Majority graph

\[
S(a) = \max \{3\}
S(b) = \max \{4\}
S(c) = \max \{3, 3\}
S(d) = \max \{4, 4\}
\]
Sequential Rules
Simple transferable vote (STV)

if there exists a candidate $c$ ranked first by a majority of votes
then $c$ wins
else Repeat

let $d$ be the candidate ranked first by the fewest voters;
eliminate $d$ from all ballots
{votes for $d$ transferred to the next best remaining candidate};
Until there exists a candidate $c$ ranked first by a majority of votes

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

Winner: $b$  
With only 3 candidates, STV coincides with plurality with runoff. 

System used in Australia, Ireland
Sequential Rules
Simple transferable vote (STV)

if there exists a candidate $c$ ranked first by a majority of votes
then $c$ wins
else Repeat

let $d$ be the candidate ranked first by the fewest voters;
eliminate $d$ from all ballots
{votes for $d$ transferred to the next best remaining candidate};
Until there exists a candidate $c$ ranked first by a majority of votes

<table>
<thead>
<tr>
<th>3</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>3</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>a</td>
</tr>
<tr>
<td>d</td>
<td>d</td>
<td>c</td>
<td>c</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>b</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>b</td>
<td>a</td>
<td>c</td>
<td>b</td>
</tr>
<tr>
<td>c</td>
<td>c</td>
<td>b</td>
<td>a</td>
<td>c</td>
<td>b</td>
<td>a</td>
<td>a</td>
</tr>
</tbody>
</table>
### Sequential Rules
**Simple transferable vote (STV)**

If there exists a candidate $c$ ranked first by a majority of votes

then $c$ wins

else Repeat

   let $d$ be the candidate ranked first by the fewest voters; 
   eliminate $d$ from all ballots 
   \{ votes for $d$ transferred to the next best remaining candidate \}; 

Until there exists a candidate $c$ ranked first by a majority of votes

<table>
<thead>
<tr>
<th></th>
<th>3</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>7</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$a$</td>
<td>$b$</td>
<td>$c$</td>
<td>$d$</td>
<td>$d$</td>
<td>$c$</td>
<td>$a$</td>
<td>$b$</td>
<td>$b$</td>
<td>$b$</td>
</tr>
<tr>
<td></td>
<td>$d$</td>
<td>$d$</td>
<td>$c$</td>
<td>$c$</td>
<td>$c$</td>
<td>$d$</td>
<td>$a$</td>
<td>$b$</td>
<td>$a$</td>
<td>$c$</td>
</tr>
<tr>
<td>$b$</td>
<td>$a$</td>
<td>$c$</td>
<td>$b$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Winner: $b$

- with only 3 candidates, STV coincides with plurality with runoff.
- system used in Australia, Ireland
Approval Voting

Here the input provided by the voters is different.

- a **profile** = a *subset of candidates* $A_i \subseteq X$ for each voter
  $P = (A_1, \ldots, A_n)$

- $S_P(x)$ = number of voters $i$ such that $x \in A_i$.

- winner: candidate maximizing $S_P$. 
A brief history
(courtesy of Jérôme Lang)

1. end of 18th century: Condorcet and Borda

2. 1951: Arrow’s theorem, birth of modern social choice theory

3. from the late 80’s on: computer scientists (and especially AI researchers) jump on board

The study of voting rules unveiled many “paradoxes”...

**Example (Saari, 1995)**

<table>
<thead>
<tr>
<th>6</th>
<th>5</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
<td>b</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>c</td>
<td>a</td>
<td>a</td>
</tr>
</tbody>
</table>

- Veto, Condorcet and Borda agree on the ranking $b > c > a$
- But plurality instead says $a > c > b$
- Other results show striking distinctions between rules, eg: No positional rule is Condorcet-consistent (Young)
An Axiomatic Approach

- Most results in (classical) social choice seek characterizations of voting rules in terms of axioms they fulfill.

- There are other ways to “rationalize” the use of certain voting rules:
  - maximum likelihood approach (there is a correct outcome, and the votes are noisy/distorted perceptions of this outcome, for a given model of noise)
  - distance-based rationalization (there is a consensus notion, and the winner is the winning candidate in the closest consensual profile, for a given notion of distance)


Sometimes *impossibility results* state that no voting rule can satisfy a given set of axioms.

- **unanimity** if \( x \succ_i y \) for every voter \( i \), then \( x \succ_P y \)
- **independence of irrelevant alternative** the social preference among \( x \) and \( y \) only depends on their relative relative ranking by every individual.
  \[
  N^P(x, y) = N^{P'}(x, y) \text{ then } x \succ_P y \iff x \succ_{P'} y
  \]
- **dictatorship** a voter \( i \) is a dictator if the function maps any profile to his vote, *i.e.* \( H : \mathcal{P}^n \rightarrow V_i \)

**Theorem (Arrow, 1951)**

*Any social welfare function for 3 or more candidates satisfying unanimity and independence must be a dictatorship.*
An example of a possibility result...

- **anonymity** does not depend on the identity of voters, i.e.
  \[ F(V_1, \ldots, V_n) = F(\pi(V_1), \ldots, \pi(V_n)) \]
- **neutrality** does not depend on the identity of candidates
- **positive responsiveness** if a candidate \( x \) is among the winners, then it should become the unique winner when some voters modify their preference and put \( x^* \) at a higher rank (without modifying the rest).

**Theorem (May, 1952)**

A voting correspondence for exactly 2 candidates satisfies anonymity, neutrality, and positive responsiveness iff it is the plurality rule (simple majority).
Another important notion is that of **strategy-proofness**.

A voting rule is strategy-proof if no voter is better-off *(i.e. prefers the new obtained winner)* misrepresenting his vote *(in any profile)*.

- **surjectivity** no candidate is discarded *(for any candidate $x$, there is a profile $P$ such that $F(P) = x$)*

**Theorem (Gibbard-Satterwaite, 1952)**

*Any voting rule for 3 or more candidates that is surjective and strategy-proof must be dictatorship.*
Plurality with runoff fails to meet positive responsiveness...

<table>
<thead>
<tr>
<th>6</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>b</td>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>c</td>
<td>b</td>
<td>a</td>
</tr>
</tbody>
</table>

1st round: b eliminated
2nd round: a elected (11/6)
Plurality with runoff fails to meet positive responsiveness...

1st round: c eliminated
2nd round: b elected (9/8)
Early motivations for computational social choice

In all these results, no consideration for computational issues

- are some rules difficult to compute?
- how about the difficulty of manipulating the election?
- how do these rules cater in distributed environment?
- what if the number of candidates is huge?
Outline of the Talk

1. Voting Rules
2. Basics of Social Choice
3. Computing Winners
4. Manipulation
5. Communication Complexity
6. Voting with Incomplete Profiles
7. Other Topics
Computing voting rules

Most voting rules can be computed in polynomial time

**Examples**:

- positional scoring rules, approval: $O(np)$
- Copeland, Simpson, STV: $O(np^2)$

But some voting rules are NP-hard.

**Reference papers**

Looking for rankings that are as “close” as possible to the preference profile and chooses the top-ranked candidates in these rankings.

- **Kemeny distance**:
  \[
  d_K(V, V') = \text{number of } (x, y) \in X^2 \text{ on which } V \text{ and } V' \text{ disagree}
  \]
  \[
  d_K(V, \langle V_1, \ldots, V_n \rangle) = \sum_{i=1, \ldots, n} d_K(V, V_i)
  \]

- **Kemeny consensus** = linear order \(\succ_P\) such that \(d_K(\succ_P, \langle V_1, \ldots, V_n \rangle)\) minimum

- **Kemeny winner** = candidate ranked first in a Kemeny consensus
A characterization of Kemeny With each profile $P$ associate the pairwise comparison matrix (recall $\#N^P(x, y)$ is the number of voters who prefer $x$ to $y$ in $P$).

Now given a ranking $R$:

$$K(R) = \sum_{x \succ_R y} \#N(x, y)$$

- If $x \succ_R y$ then this corresponds to $\#N(x, y)$ agreements (and $\#N(y, x)$ disagreements)

- $P^*$ is a Kemeny consensus iff $K(P^*)$ is maximum.

<table>
<thead>
<tr>
<th>4 voters</th>
<th>3 voters</th>
<th>2 voters</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$b$</td>
<td>$c$</td>
</tr>
<tr>
<td>$b$</td>
<td>$c$</td>
<td>$a$</td>
</tr>
<tr>
<td>$c$</td>
<td>$a$</td>
<td>$b$</td>
</tr>
</tbody>
</table>

Find the Kemeny winner(s).
## Hard rules

### Kemeny

<table>
<thead>
<tr>
<th></th>
<th>4 voters</th>
<th>3 voters</th>
<th>2 voters</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(b)</td>
<td>(c)</td>
<td></td>
</tr>
<tr>
<td>(b)</td>
<td>(c)</td>
<td>(a)</td>
<td></td>
</tr>
<tr>
<td>(c)</td>
<td>(a)</td>
<td>(b)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(N)</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(-)</td>
<td>(6)</td>
<td>(4)</td>
</tr>
<tr>
<td>(b)</td>
<td>(3)</td>
<td>(-)</td>
<td>(7)</td>
</tr>
<tr>
<td>(c)</td>
<td>(5)</td>
<td>(2)</td>
<td>(-)</td>
</tr>
</tbody>
</table>

### Kemeny scores

\[
\begin{array}{ccc}
abc & acb & bac & bca & cab & cba \\
17 & 12 & 14 & 15 & 13 & 10 \\
\end{array}
\]

**Kemeny consensus**: \(abc\); **Kemeny winner**: \(a\)

- this naive approach yields \(O(p!p^2n)\)
early results: Kemeny is NP-hard (Orlin, 81; Bartholdi et al., 89; Hudry, 89)

deciding whether a candidate is a Kemeny winner is not even in NP, but higher up

many works on approximation
Technique of **Local Kemenization**:

1. generate an initial ranking $R$ [w.l.o.g., $R = x_1 \succ \ldots \succ x_m$];
2. for $k := 2$ to $m$ do
3.  for $j := k - 1$ downto 1 do
4.     if $x_{j+1}$ beats $x_j$ majoritywise
5.     then swap $x_j$ and $x_{j+1}$ in $R$.
6. return $R$.

- computable in polynomial time (provided the initial ranking is computable in polynomial time)

Hard rules
Kemeny

- Used in meta-search engines (in that case, rankings are likely to be partial, because of limited size)
- The result may be arbitrary far from optimal

\[
\begin{array}{ccc}
1 & 1 & 2 \\
a & b & c \\
b & c & a \\
\end{array}
\]

The ranking \( a \succ b \succ c \) is locally optimal, but \( c \succ b \succ a \) is the consensus ranking.

- the resulting ranking \( R \) satisfies a property of “generalized Condorcet-consistency” (which turns out to be very appropriate for the problem at hand, so domain-specific criterion are also to consider)
Other examples of rules difficult to compute:

**Dodgson (＝ Lewis Carrol) rule** for each candidate $c$, compute $D(c)$ the number of adjacent swaps required to turn it into a Condorcet winner. Pick the candidate minimizing $D(c)$.

- Deciding whether a designated candidate $x$ is a Dodgson winner is NP-hard, not in NP, but higher up in the hierarchy. So even verifying is not easy.
Other examples of rules difficult to compute:

**Dodgson (= Lewis Carrol) rule** for each candidate $c$, compute $D(c)$ the number of adjacent swaps required to turn it into a Condorcet winner. Pick the candidate minimizing $D(c)$.

- Deciding whether a designated candidate $x$ is a Dodgson winner is NP-hard, not in NP, but higher up in the hierarchy. So even verifying is not easy.

**Young rule** for each candidate $c$, compute $Y(c)$ the smallest number of voters that we need to remove to turn it into a Condorcet winner. Pick the candidate minimizing $Y(c)$.

- Deciding whether a designated candidate $x$ is a Young winner is NP-hard, not in NP, but higher up in the hierarchy.


Outline of the Talk

1. Voting Rules
2. Basics of Social Choice
3. Computing Winners
4. Manipulation
5. Communication Complexity
6. Voting with Incomplete Profiles
7. Other Topics
The many facets of manipulation

Remember Gerrymandering. Producing automatically “fair” redistricting was an early motivation to consider algorithmic issues in voting.

▶ (Garfinkel & Nemhauser, 70) : early algorithms.
▶ (Altman, 1997) : “fair” redistricting is NP-hard.

Survey paper

Ricca, Scozzari & Simeone. Political districting : from classical models to recent approaches. 4OR, 2011.
The many facets of manipulation

Remember Gerrymandering. Producing automatically “fair” redistricting was an early motivation to consider algorithmic issues in voting.

- (Garfinkel & Nemhauser, 70) : early algorithms.
- (Altman, 1997) : “fair” redistricting is NP-hard.

Survey paper

Ricca, Scozzari & Simeone. Political districting : from classical models to recent approaches. 4OR, 2011.

In our context, no districts. But many different variants of the problem though :

- Who wants to manipulate ? (a single voter, a group/coalition of voters, the chair of the election)
- What kind of manipulation is allowed ? (modifying only his own vote, buying to get the others to modify their votes)
Recall the Gibbard-Satterwaite Theorem...
If manipulation is computationally prohibitive then this may be a good news.
However always bear in mind that this is only a worst-case concept (so manipulation may be easy on most instances...)

Survey paper

Complexity of manipulation

Two types of manipulation can be distinguished:

- **constructive manipulation existence**: Given a voting rule \( r \), a set of \( p \) candidates \( \mathcal{X} \), a candidate \( x \in \mathcal{X} \), and the votes of voters \( 1, \ldots, k < n \) Question is there a way for voters \( k + 1, \ldots, n \) to cast their votes such that \( x \) is elected?

- **destructive manipulation existence**: Given a voting rule \( r \), a set of \( p \) candidates \( \mathcal{X} \), a candidate \( x \in \mathcal{X} \), and the votes of voters \( 1, \ldots, k < n \) Question is there a way for voters \( k + 1, \ldots, n \) to cast their votes such that \( x \) is *not* elected?
Manipulating Borda
The single voter case

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>a</td>
<td>e</td>
<td>c</td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>c</td>
<td>d</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>e</td>
<td>c</td>
<td>d</td>
<td>e</td>
</tr>
</tbody>
</table>

Current Borda scores:
- a: 10
- b: 10
- c: 8
- d: 7
- e: 5

▶ Is there a constructive manipulation for a? for b? for c? for d? for e?
Manipulating Borda
The single voter case

Current Borda scores:
- a: 10
- b: 10
- c: 8
- d: 7
- e: 5

▶ Is there a constructive manipulation for a and for b? Obviously yes.
Manipulating Borda
The single voter case

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>a</td>
<td>e</td>
<td>c</td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>c</td>
<td>d</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>e</td>
<td>c</td>
<td>d</td>
<td>e</td>
</tr>
</tbody>
</table>

Current Borda scores:
- a: 10
- b: 10
- c: 8
- d: 7
- e: 5

Is there a constructive manipulation for c?

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>a</td>
<td>e</td>
<td>c</td>
<td>e</td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td>a</td>
<td>b</td>
<td>d</td>
</tr>
<tr>
<td>c</td>
<td>d</td>
<td>b</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>e</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>b</td>
</tr>
</tbody>
</table>

Borda scores:
- a: 10 + 1 = 11
- b: 10 + 0 = 10
- c: 8 + 4 = 12
- d: 7 + 2 = 9
- e: 5 + 3 = 8

▶ yes
Manipulating Borda
The single voter case

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>a</td>
<td>e</td>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>c</td>
<td>d</td>
<td>b</td>
<td>a</td>
<td>e</td>
</tr>
<tr>
<td>e</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>b</td>
</tr>
</tbody>
</table>

Current Borda scores:
- a: 10
- b: 10
- c: 8
- d: 7
- e: 5

▶ Is there a constructive manipulation for d?

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>a</td>
<td>e</td>
<td>d</td>
<td>c</td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td>a</td>
<td>c</td>
<td>b</td>
</tr>
<tr>
<td>c</td>
<td>d</td>
<td>b</td>
<td>a</td>
<td>e</td>
</tr>
<tr>
<td>e</td>
<td>c</td>
<td>d</td>
<td>b</td>
<td>a</td>
</tr>
</tbody>
</table>

Borda scores:
- a: 10 + 1 = 11
- b: 10 + 0 = 10
- c: 8 + 2 = 10
- d: 7 + 4 = 11
- e: 5 + 3 = 8

▶ the answer depends on the tie-breaking priority of d.
Manipulating Borda

The single voter case

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>a</td>
<td>e</td>
<td>c</td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>c</td>
<td>d</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>e</td>
<td>c</td>
<td>d</td>
<td>e</td>
</tr>
</tbody>
</table>

Current Borda scores:
- a: 10
- b: 10
- c: 8
- d: 7
- e: 5

Is there a constructive manipulation for e? obviously not.
Complexity of manipulation
Manipulating the Borda rule by a single voter

Without loss of generality:

- \( P \) profile (without the manipulating voter)
- \( x_1 \) candidate that the voter wants to see winning
- \( x_2, \ldots, x_m \) other candidates, ranked by decreasing Borda score w.r.t. the current profile

**Algorithm**: place \( x_1 \) on top, then \( x_m \) in second position, then \( x_{m-1}, \ldots, \), and finally \( x_2 \) in the bottom position.

If \( x_1 \) does not become a winner then there exists no manipulation for \( x \).

- thus for Borda, constructive manipulation existence by one voter is in P. (Bartholdi, Tovey & Trick, 89).
- manipulation by coalitions of more than one voter: NP-hardness recently solved (Betzler et al., 2011) and (Davies et al., 2011)
- some rules are hard to manipulate even for a single voter, for instance the STV rule (Bartholdi & Orlin, 91)
- some empirical works on manipulation as well (Walsh et al. 2010)
Borda + tie-breaking priority $a > b > c > d > e$.
Current Borda scores:

$a : 12, b : 10, c : 9, d : 9, e : 4, f : 1$

Is there a constructive manipulation by two voters for $e$?
Complexity of (unweighted) manipulation

From Xia et al. (09):

<table>
<thead>
<tr>
<th>Number of manipulators</th>
<th>1</th>
<th>at least 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copeland</td>
<td>P (1)</td>
<td>NP-complete (2)</td>
</tr>
<tr>
<td>STV</td>
<td>NP-complete (3)</td>
<td>NP-complete (3)</td>
</tr>
<tr>
<td>veto</td>
<td>P (4)</td>
<td>P (4)</td>
</tr>
<tr>
<td>Simpson</td>
<td>P (1)</td>
<td>NP-complete (6)</td>
</tr>
<tr>
<td>Borda</td>
<td>P (1)</td>
<td>NP-complete (7, 8)</td>
</tr>
</tbody>
</table>

(1) Bartholdi et al.; (2) Falisezwski et al.; (3) Bartholdi and Orlin; (4) Zuckerman et al.; (7) Betzler et al. (8) Davies et al.
The main types of control are adding/deleting voters/candidates.

With respect to a given type of control, we say that a voting rule is:

- **immune** if this control can never turn a non-winning candidate into a winning one;
- **resistant** if it is not immune but it is difficult (i.e. NP-hard) to decide whether the outcome can be obtained;
- **vulnerable** if it not immune and, furthermore, easy.

From (Bartholdi et al., 92) and (Trick, 09):

<table>
<thead>
<tr>
<th>Control by</th>
<th>Plurality</th>
<th>Condorcet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adding candidates</td>
<td>resistant</td>
<td>immune</td>
</tr>
<tr>
<td>Deleting candidates</td>
<td>resistant</td>
<td>vulnerable</td>
</tr>
<tr>
<td>Adding voters</td>
<td>vulnerable</td>
<td>resistant</td>
</tr>
<tr>
<td>Deleting voters</td>
<td>vulnerable</td>
<td>resistant</td>
</tr>
</tbody>
</table>
Outline of the Talk

1. Voting Rules
2. Basics of Social Choice
3. Computing Winners
4. Manipulation
5. Communication Complexity
6. Voting with Incomplete Profiles
7. Other Topics
Communication Complexity

- important to know the amount of information that needs to be exchanged to compute the outcome
- no concern regarding the computational power of agents here
- a naive universal protocol:
  1. each agent reports his own vote to the center ($n \log p!$ bits)
  2. the center sends back the result (name of the winner) ($n \log p$ bits)
- for specific rules we may design more clever protocols
- specific protocols provide upper bounds on the communication complexity of the voting rule

A possible protocol:

1. **Step 1**
   - Voters send the name of their most preferred candidate to the central authority
   - Communication: $n \log p$ bits

2. **Step 2**
   - The central authority sends the names of the two finalists to the voters
   - Communication: $2n \log p$ bits

3. **Step 3**
   - Voters send the name of their preferred finalist to the central authority
   - Communication: $n$ bits

**Total** $n(3\log p + 1)$ bits (in the worst case)

- The communication complexity of plurality with runoff is in $O(n \cdot \log p)$. 
Upper Bounds
Single Transferable Vote (STV)

A slightly more intricated protocol...

**step 1** voters send their most preferred candidate to the central authority \( (C) \)
\[ \leftrightarrow n \log p \text{ bits} \]

**step 2** let \( x \) be the candidate to be eliminated. All voters who had \( x \) ranked first receive a message from \( C \) asking them to send the name of their next preferred candidate. There were at most \( \frac{n}{p} \) such voters
\[ \leftrightarrow \frac{n}{p} \log p \text{ bits} \]

**step 3** similarly with the new candidate \( y \) to be eliminated. At most \( \frac{n}{p-1} \) voters voted for \( y \)
\[ \leftrightarrow \frac{n}{p-1} \log p \text{ bits} \]

etc.

**total** \[ \leq n \log p \left( 1 + \frac{1}{p} + \frac{1}{p-1} + \ldots + \frac{1}{2} \right) = \mathcal{O}(n.(\log p)^2). \]
Basic communication complexity setting

A set of $n$ agents have to compute a function $f(x_1, \ldots, x_n)$ given that the input is distributed among the agents ($x_1$ privately known from agent 1, etc.)

- **Protocols**: Specify a communication action by the agents, given its (private) input and the bits exchanged so far.

- **Useful tree representation** where each node is labelled by either agent $a$ or agent $b$ (case of two agents), with a function specifying whether to walk left (0) or right (1) depending on its private input.

Protocols illustrated

\[
\begin{align*}
\text{a}(x_0) &= 0 \\
\text{a}(x_1) &= 1 \\
\text{a}(x_2) &= 1 \\
\text{a}(x_3) &= 0
\end{align*}
\]

\[
\begin{align*}
\text{b}(y_0) &= 0 \\
\text{b}(y_1) &= 0 \\
\text{b}(y_2) &= 0 \\
\text{b}(y_3) &= 1
\end{align*}
\]

\[
\begin{array}{c|cccc}
& y_0 & y_1 & y_2 & y_3 \\
\hline
x_0 & 0 & 0 & 0 & 1 \\
x_1 & 0 & 0 & 0 & 0 \\
x_2 & 0 & 0 & 0 & 0 \\
x_3 & 1 & 1 & 1 & 0 \\
\end{array}
\]
Protocols illustrated

\[ b(y_0) = 0 \]
\[ b(y_1) = 0 \]
\[ b(y_2) = 0 \]
\[ b(y_3) = 1 \]

\[ a(x_0) = 0 \]
\[ a(x_1) = 0 \]
\[ a(x_2) = 0 \]
\[ a(x_3) = 1 \]

<table>
<thead>
<tr>
<th></th>
<th>(y_0)</th>
<th>(y_1)</th>
<th>(y_2)</th>
<th>(y_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x_0)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(x_1)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(x_2)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(x_3)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Cost of protocols

The **cost of a protocol** is the number of bits exchanged (in the worst case), i.e. the height of the tree.

On our example, the “best” cost is the second one (cost 2 vs. 3 for the first one).

The **communication complexity** of a function $f$ is the minimum cost of $\mathcal{P}$ among all protocols $\mathcal{P}$ that compute $f$.

But how do we know that there is no better protocol?

- communication complexity offers a bunch of techniques to prove lower bounds
- one of them is the **fooling set** technique
Fooling sets

Observe that the protocols, as described, in fact partition the matrix of inputs into **monochromatic** (same output) rectangles

<table>
<thead>
<tr>
<th></th>
<th>y0</th>
<th>y1</th>
<th>y2</th>
<th>y3</th>
</tr>
</thead>
<tbody>
<tr>
<td>x0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>x1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

⇒ 5 monochromatic rectangles

► the number of leaves is the number of rectangles
► hence the cost of protocol must be at least the \( \log(\#\text{rectangles}) \)
► if we find a large number of inputs such that no two of them can be in the same rectangle, the number of rectangles must be large as well.

► when two input pairs \((x_1, y_1)\) and \((x_2, y_2)\) are in the same monochromatic rectangle, so do \((x_1, y_2)\) and \((x_2, y_1)\)

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c}
& y0 & y1 & y2 & y3 \\
\hline
x0 & 0 & 0 & 0 & 1 \\
x1 & 0 & 0 & 0 & 0 \\
x2 & 0 & 0 & 0 & 0 \\
x3 & 1 & 1 & 1 & 0 \\
\end{array}
\]

Key result (Yao,1979) : CC is at least \( \log(\#\text{fooling set}) \)
In our context, we have:

- \( f \) is the voting rule
- \( x_i \) is the ballot of voter \( i \)
- we are interested in a distinguished candidate \( a \), so \( f \) returns 1 if \( a \) wins, and 0 otherwise

A fooling set is then a set of profiles \( P_i \) such that:

1. there exists a candidate \( c \) such that \( r(P^i) = c \)
2. for any pair \((i, j) \) \((i \neq j)\), there exists \((m_1, m_2, \ldots, m_n) \in \{i, j\}^n\) such that \( r(v_1^{m_1}, v_2^{m_2}, \ldots, v_n^{m_n}) \neq c \)

\( \mapsto \) we can “mix” the profiles by picking votes either in \( P^i \) or \( P^j \) and fool the function.
Example : Lower bound for the Borda rule
[Conitzer & Sandholm, EC05]

We note $p' = p - 2$ and $n' = (n - 2)/4$, $\pi$ an arbitrary permutation of candidates $\mathcal{X} \setminus \{a, b\}$ and $\overline{\pi}$ the “mirror” permutation.

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & \cdots & n-1 & n \\
\hline
a & a & \overline{\pi} & \overline{\pi} & \cdots & a & \overline{\pi} \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
b & b & \vdots & \vdots & \ddots & b & \vdots \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\pi & \pi & \vdots & \vdots & \ddots & \pi & \vdots \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\vdots & \vdots & \overline{\pi} & \overline{\pi} & \cdots & \overline{\pi} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\vdots & \vdots & b & b & \cdots & a & \vdots \\
\pi & \pi & a & a & \cdots & \pi & b
\end{array}
\]

\[\Rightarrow (p')^{n'} \text{ such profiles}\]
Example: Lower bound for the Borda rule
[Conitzer & Sandholm, EC05]

We note $p' = p - 2$ and $n' = (n - 2)/4$, $\pi$ an arbitrary permutation of candidates $\mathcal{X} \setminus \{a, b\}$ and $\overline{\pi}$ the “mirror” permutation.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>⋮</th>
<th>$n - 1$</th>
<th>$n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>$\pi$</td>
<td>$\pi$</td>
<td>⋮</td>
<td>$a$</td>
<td>$\pi$</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>⋮</td>
<td>⋮</td>
<td>⋮</td>
<td>⋮</td>
<td>⋮</td>
<td></td>
</tr>
<tr>
<td>$\pi$</td>
<td>$\pi$</td>
<td>⋮</td>
<td>⋮</td>
<td>$\pi$</td>
<td>⋮</td>
<td>$\pi$</td>
<td></td>
</tr>
<tr>
<td>⋮</td>
<td>⋮</td>
<td>$\overline{\pi}$</td>
<td>$\overline{\pi}$</td>
<td>⋮</td>
<td>⋮</td>
<td>$\overline{\pi}$</td>
<td></td>
</tr>
<tr>
<td>⋮</td>
<td>⋮</td>
<td>b</td>
<td>b</td>
<td>⋮</td>
<td>a</td>
<td>⋮</td>
<td></td>
</tr>
<tr>
<td>$\pi$</td>
<td>$\pi$</td>
<td>a</td>
<td>a</td>
<td>⋮</td>
<td>$\pi$</td>
<td>b</td>
<td></td>
</tr>
</tbody>
</table>

1. Does $a$ wins in any such profile?

Observe that $a$ is 1 point ahead of any other candidate (thanks to voter $n$)
Example: Lower bound for the Borda rule
[Conitzer & Sandholm, EC05]

We note $p' = p - 2$ and $n' = (n - 2)/4$, $\pi$ an arbitrary permutation of candidates $\mathcal{X} \setminus \{a, b\}$ and $\overline{\pi}$ the “mirror” permutation.

\begin{array}{ccccccc}
1 & 2 & 3 & 4 & \cdots & n-1 & n \\
\hline
a & a & \overline{\pi} & \overline{\pi} & \cdots & a & \overline{\pi} \\
\hline
b & b & \vdots & \vdots & & b & \vdots \\
\hline
\pi & \pi & \vdots & \vdots & & \pi & \vdots \\
\vdots & \vdots & \overline{\pi} & \overline{\pi} & \vdots & \overline{\pi} \\
\vdots & \vdots & b & b & \vdots & a \\
\hline
\pi & \pi & a & a & \cdots & \pi & b \\
\end{array}

1. Does $a$ wins in any such profile?

Observe that $a$ is 1 point ahead of any other candidate (thanks to voter $n$)

2. Is it fooling?

Take two profiles $P_1$ and $P_2$, for at least one voter $i \in \{1, \ldots, n'\}$ the vote differs. Thus at least one candidate $c \not\in \{a, b\}$ must be ranked higher in $P_1$ than $P_2$. Mix profiles by picking votes $4i-3$ and $4i - 2$ from $P_1$ and the rest from $P_2$. Now $c$ get 2 additional points and wins.
Outline of the Talk

1. Voting Rules
2. Basics of Social Choice
3. Computing Winners
4. Manipulation
5. Communication Complexity
6. Voting with Incomplete Profiles
7. Other Topics
Voting with Incomplete Profiles

There are many cases where profiles can be incomplete:

(i) cannot compare candidates (intrinsic incompleteness)
(ii) there are too many candidates to be ranked
(iii) messages may be lost, delayed, or faulty

When profiles are incomplete, one may either rely on:

- further communication to elicitate the (relevant) missing information
- computation of possible winner(s), i.e. candidates who win in at least one completion of the profile
Possible and necessary winners

- For each voter: $P_i$ is a partial order on the set of candidates.
- $P = \langle P_1, \ldots, P_n \rangle$ incomplete profile
- **Completion** of $P$: full profile $T = \langle T_1, \ldots, T_n \rangle$ of $P$, where each $T_i$ is a linear ranking extending $P_i$.
- $r$ voting rule

- $c$ is a **possible winner** if there exists a completion of $P$ in which $c$ is elected.
- $c$ is a **necessary winner** if $c$ is elected in every completion of $P$.

Possible and necessary winners

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$a \succ b$, $a \succ c$</td>
<td>$b \succ a$</td>
<td>$c \succ a \succ b$</td>
</tr>
<tr>
<td>abc</td>
<td>cba</td>
<td>cab</td>
</tr>
<tr>
<td>abc</td>
<td>bca</td>
<td>cab</td>
</tr>
<tr>
<td>abc</td>
<td>bac</td>
<td>cab</td>
</tr>
<tr>
<td>acb</td>
<td>cba</td>
<td>cab</td>
</tr>
<tr>
<td>acb</td>
<td>bca</td>
<td>cab</td>
</tr>
<tr>
<td>acb</td>
<td>bac</td>
<td>cab</td>
</tr>
</tbody>
</table>

possible plurality $b \succ a \succ c$-winners: $\{b, c\}$. 

possible winners for plurality with tie-breaking $b \succ a \succ c$

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>c</td>
</tr>
<tr>
<td></td>
<td>b</td>
</tr>
<tr>
<td></td>
<td>b</td>
</tr>
<tr>
<td></td>
<td>c</td>
</tr>
<tr>
<td></td>
<td>b</td>
</tr>
<tr>
<td></td>
<td>c</td>
</tr>
</tbody>
</table>
Missing Voters and Missing Candidates

In his general version, the problem of voting under incomplete preferences makes no assumption on incompleteness:
But two specific sub-cases of the problem are natural.

<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>b</td>
<td>c</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td>a</td>
<td>a</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>a</td>
<td>c</td>
<td>b</td>
<td>c</td>
<td></td>
</tr>
</tbody>
</table>

Missing voters:

<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>b</td>
<td>c</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td>a</td>
<td>a</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>a</td>
<td>c</td>
<td>b</td>
<td>c</td>
<td></td>
</tr>
</tbody>
</table>

Missing candidates:

<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>b</td>
<td>c</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td>a</td>
<td>a</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>a</td>
<td>c</td>
<td>b</td>
<td>c</td>
<td></td>
</tr>
</tbody>
</table>

▶ Observe that the possible winner problem with missing voters exactly correspond the coalitional manipulation problem.
Unknown number of missing voters: how to store the current profile?

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

4 5

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td></td>
</tr>
</tbody>
</table>

The winner is: x
**Missing Voters**

Compiling Profiles

---

**Unknown number of missing voters:** how to store the **current** profile?

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

**Compilation function**

```
   1  2  3  ...  
  a b b 
  b c a 
  c a c 

   4  5  
  c a 
  a b 
  b c 
```

```
10011000101
```

**The winner is:** x

---

```
   4  5  
  c a 
  a b 
  b c 
```

```
10011000101
```

**The winner is:** x

---

**same winner**
We are after the best compilation functions for each voting rule. To start with, for any anonymous voting rule, compiling the profile into the corresponding voting situation is possible:

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>b</td>
<td>c</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td>a</td>
<td>a</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>a</td>
<td>c</td>
<td>b</td>
<td>c</td>
<td></td>
</tr>
</tbody>
</table>

Voting situation:

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

Hence the compilation requires at most $\min(n \log p!, p! \log n)$. 
We are after the best compilation functions for each voting rule. To start with, for any anonymous voting rule, compiling the profile into the corresponding voting situation is possible:

Profile:

<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>b</td>
<td>c</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td>a</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>c</td>
<td>a</td>
<td>c</td>
<td>b</td>
<td>c</td>
</tr>
</tbody>
</table>

Voting situation:

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

Hence the compilation requires at most \( \min(n \log p!, p! \log n) \).

- Very efficient when \( n \gg p \). Eg. \( n = 4703 \) and \( p = 4 \) we get \( \min(4703 \log 24, 24 \log 4703) \) so 312 bits vs. 23515 bits.
Intuitively, for specific voting rules one can get much better compilations, eg. for plurality just compile the score yields $p \log n$.

But these are upper bounds: how do we know that no better compilation is possible by a very smart guy?

Lower bounds: again, borrow notions from communication complexity.

In fact, the problem can be seen as a “one-round” communication complexity problem (the center must send the relevant information in one single message)
two profiles are equivalent for a voting rule if they return the same winner for any possible completion.

the key is to characterize equivalence classes for each rules, and enumerate them (not always easy...).

the compilation complexity is given by taking the log of this number.

\[
\begin{array}{|c|c|c|}
\hline
\text{Voting rule} & \text{Characterization of equiv.} & \text{Compilation complexity} \\
\hline
\text{Any voting rule} & \text{same profiles} & O(np \log p) \\
\text{Anonymous} & \text{same voting situations} & O(p! \log n) \\
\text{STV} & \text{for all } Z \subseteq C \text{ and } x \not\in Z, \text{\[score_P(x, P - Z) = score_P(x, Q - Z)\]} & \Omega(2^p \log n) \text{ or } O(p2^p \log n) \\
\text{Plurality/runoff} & M_P = M_Q \text{ and } \text{\[score_P(x, P) = score_P(x, Q)\]} & \Theta(p^2 \log n) \\
\text{Cond. WMG} & M_P = M_Q & O(p^2 \log n) \\
\text{Borda} & \text{\[score_B(x, P) = score_B(x, Q)\]} & \Theta(p \log np) \\
\text{Plurality} & \text{\[score_P(x, P) = score_P(x, Q)\]} & \Theta(p \log(1 + \frac{n}{p}) + n \log(1 + \frac{p}{n})) \\
\hline
\end{array}
\]


Recall that here partial votes are simply linear orders on a subset of candidates ($k$ missing candidates).

**Example** Job assignment decision with 4 valid applications and 2 pending verifications.

Given a voting situation $\pi$ and a voting rules $r$, $x \in X$ is a possible winner (wrt $\pi$ and $r$) if there is a completion, a profile $P$ extending $P_X$, st. $r(P) = x$

Note that the necessary winner problem is not very relevant here, any new candidate being (under mild conditions) a possible winner.

Study the possible winner problem with new candidates focusing on scoring rules $\langle s_1, \ldots, s_p \rangle$ ($s_i \geq s_{i+1}$ and $s_1 > s_p$).

Missing Candidates
An example with plurality

Example (Plurality, 1 new candidate)

1: \( a \succ d \succ c \succ b \)
2: \( a \succ b \succ c \succ d \)
3: \( a \succ d \succ c \succ b \)
4: \( d \succ a \succ c \succ b \)
5: \( b \succ a \succ c \succ d \)
6: \( b \succ d \succ a \succ c \)
7: \( c \succ d \succ a \succ b \)
8: \( c \succ b \succ d \succ a \)

Tie-breaking: \( a > b > c > d > y \)

Plurality scores:
\( s(a) = 3, \ s(b) = 2, \ s(c) = 2, \ s(d) = 1 \)

Who are the possible winners?
certainly \( a \) is...
### Example (Plurality, 1 new candidate)

<table>
<thead>
<tr>
<th>Rank</th>
<th>Preference</th>
<th>Plurality Scores</th>
<th>Tie-breaking</th>
<th>Possible Winners</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( a \succ d \succ c \succ b \succ y )</td>
<td>( s(a) = 3, s(b) = 2, s(c) = 2, s(d) = 1 )</td>
<td>( a &gt; b &gt; c &gt; d &gt; y )</td>
<td>( b ) is as well...</td>
</tr>
<tr>
<td>2</td>
<td>( y \succ a \succ b \succ c \succ d )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>( y \succ a \succ d \succ c \succ b )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>( d \succ a \succ c \succ b \succ y )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>( b \succ a \succ c \succ d \succ y )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>( b \succ d \succ a \succ c \succ y )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>( c \succ d \succ a \succ b \succ y )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>( c \succ b \succ d \succ a \succ y )</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Example (Plurality, 1 new candidate)

<table>
<thead>
<tr>
<th>Rank</th>
<th>Candidate</th>
<th>Plurality scores</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>y</td>
<td>( s(y) = 1 )</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>( s(a) = 3 )</td>
</tr>
<tr>
<td>3</td>
<td>b</td>
<td>( s(b) = 2 )</td>
</tr>
<tr>
<td>4</td>
<td>c</td>
<td>( s(c) = 2 )</td>
</tr>
<tr>
<td>5</td>
<td>d</td>
<td>( s(d) = 1 )</td>
</tr>
</tbody>
</table>

### Tie-breaking:
\( a > b > c > d > y \)

Who are the possible winners?

- a and b are possible winners.
- c is not a possible winner.
Example (Plurality, 2 new candidates)

1: \( a \succ d \succ c \succ b \succ y_1 \succ y_2 \)
2: \( y_1 \succ a \succ b \succ c \succ d \succ y_2 \)
3: \( y_2 \succ a \succ d \succ c \succ b \succ y_1 \)
4: \( d \succ a \succ c \succ b \succ y_1 \succ y_2 \)
5: \( y_1 \succ b \succ a \succ c \succ d \succ y_2 \)
6: \( b \succ d \succ a \succ c \succ y_1 \succ y_2 \)
7: \( c \succ d \succ a \succ b \succ y_1 \succ y_2 \)
8: \( c \succ b \succ d \succ a \succ y_1 \succ y_2 \)

Tie-breaking: \( a \succ b \succ c \succ d \succ y \)

Plurality scores:
\( s(a) = 3, s(b) = 2, s(c) = 2, s(d) = 1 \)

Who are the possible winners? now \( c \) is.
Plurality : an easy case

The general condition is easy to state. Intuitively :

- each new candidate can be placed at the top to decrease the score of a candidate;
- for each candidate with a higher score than $x$ we must put the new candidate on top a number of times equal to the difference of scores ($+1$ if that candidate has priority in the tie-breaking rule);
- the score of the new candidate must not be higher (or indeed equal if the new candidate has priority) than the current score of $x$.

Generalizes to $k$ new candidates.

$$\text{top}(P_X, x) \geq \frac{1}{k} \sum_{z \in X} \max(0, \text{top}(P_X, z) - \text{top}(P_X, x))$$
Consider the scoring vector $\langle p - 1, p - 2, \ldots, 0 \rangle$. For a given candidate $x$, the best situation is that the new candidates $y_i$ are placed right after $x$ in the profile.

$$\langle 4, 3, 2, 1, 0 \rangle$$

$a \succ x \succ y \succ b \succ c$
Consider the scoring vector $\langle p - 1, p - 2, \ldots, 0 \rangle$.
For a given candidate $x$, the best situation is that the new candidates $y_i$ are placed right after $x$ in the profile.

$$\langle 4, 3, 2, 1, 0 \rangle$$

$$a \succ x \succ y \succ b \succ c$$

Holds in general for rules where $\forall i : (s_i - s_{i+1}) \leq (s_{i+1} - s_{i+2})$
Consider the scoring vector \( \langle p - 1, p - 2, \ldots, 0 \rangle \).
For a given candidate \( x \), the best situation is that the new candidates \( y_i \) are placed right after \( x \) in the profile.

\[
\langle 4, 3, 2, 1, 0 \rangle
\]

\[
a \succ x \succ y \succ b \succ c
\]

Holds in general for rules where \( \forall i : (s_i - s_{i+1}) \leq (s_{i+1} - s_{i+2}) \)
But the property doesn’t hold if the score vector is “convex” :

\[
\langle 10, 3, 2, 1, 0 \rangle
\]

\[
a \succ x \succ y \succ b \succ c
\]
Consider the scoring vector $\langle p - 1, p - 2, \ldots, 0 \rangle$. For a given candidate $x$, the best situation is that the new candidates $y_i$ are placed right after $x$ in the profile.

$$\langle 4, 3, 2, 1, 0 \rangle$$

$a \succ x \succ y \succ b \succ c$

Holds in general for rules where $\forall i : (s_i - s_{i+1}) \leq (s_{i+1} - s_{i+2})$

But the property doesn’t hold if the score vector is “convex”:

$$\langle 10, 3, 2, 1, 0 \rangle$$

$y \succ a \succ x \succ b \succ c$

May be good to put $y$ above $x$ here (because $a$ loses 7 points)...
This means that the condition is also easy to state. A candidate can only gain points against another candidate when it is above. Let $N(P_X, x, z)$ the number of times $x$ this happens.

$$k \geq \max_{z \in X \setminus \{x\}} \frac{s(P_X, z) - s(P_X, x)}{N(P_X, x, z)}$$
Example (Borda)

1: \( a \succ d \succ c \succ b \) \hspace{1cm} \text{Borda scores :} \hspace{1cm} s(a) = 15, s(b) = 10, s(c) = 11, s(d) = 12
2: \( a \succ b \succ c \succ d \)
3: \( a \succ d \succ c \succ b \) \hspace{1cm} \delta(b, a) = (15 - 10)/3 = 5/3
4: \( d \succ a \succ c \succ b \) \hspace{1cm} \delta(b, c) = (11 - 10)/3 = 1/3
5: \( b \succ a \succ c \succ d \) \hspace{1cm} \delta(b, d) = (12 - 10)/4 = 1/2
6: \( b \succ d \succ a \succ c \) \hspace{1cm} \text{Hence 2 new candidates are required for } b.
7: \( c \succ d \succ a \succ b \) \hspace{1cm} \text{And the possible winners are...}
8: \( c \succ b \succ d \succ a \) \hspace{1cm} \text{with 1 new candidate } d
\hspace{1cm} \text{with 2 new candidates } b \text{ and } c
\( k \)-approval

\[ \langle 1, \ldots, 1, 0, \ldots, 0 \rangle \]

For one candidate, the condition can be easily stated. Intuitively:

- Only candidates lying in the last position of the approval set can be "pushed away";
- Candidates with a higher score than \( x \) must appear in the last approved position sufficiently many times;
- Overall, the score of the new candidate must not exceed the current score of \( x \).
For one candidate, the condition can be easily stated. Intuitively:

- Only candidates lying in the last position of the approval set can be "pushed away";
- Candidates with a higher score than \( x \) must appear in the last approved position sufficiently many times;
- Overall, the score of the new candidate must not exceed the current score of \( x \).

Generalizes to more than one new candidate?
4-approval is hard

**Theorem**

*Deciding if $x$ is a possible winner for 4-approval w.r.t. the addition of 3 candidates is NP-complete*

**Proof (sketch)**: Reduction to the perfect 3D-matching problem.

Given: a collection of triples $S_1 \cup \ldots \cup S_n$ where $S_i = (a_i, b_i, c_i)$, find a perfect matching if there exists one.
4-approval is hard

The voting profile is build such that:

- \( \text{score}(S_i) = 1 \), \( \text{score}(x) = n \)
- \( \text{score}(a_i) = \text{score}(b_i) = \text{score}(c_i) = n + 1 \)

To make \( x \) win, add 3 candidates \( w_1, w_2, w_3 \) such that:

- These candidates must appear at most \( n \) times (otherwise they win)
- They must lower the score of each \( a_i, b_i, c_i \). e.g. \( S_1 \succ a_1 \succ b_1 \succ c_1 \) becomes \( S_1 \succ w_1 \succ w_2 \succ w_3 \)
- The only way to do this is to remove from top candidates each \( a_i, b_i, c_i \) exactly once, which is 3DM.
Outline of the Talk

1. Voting Rules
2. Basics of Social Choice
3. Computing Winners
4. Manipulation
5. Communication Complexity
6. Voting with Incomplete Profiles
7. Other Topics
Other Topics

- **Axiomatizing search engine**
  Applies the axiomatic approach to search engines. Interestingly, in that case, voters and candidates coincide

- **Judgement aggregation**
  The aim is to aggregate judgments of agents on propositional formulae. The literature was triggered by the so-called doctrinal paradox:

\[
\begin{array}{cccc}
p & q & p \land q & r \\
1 & 1 & 1 & 1 \\
1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 \\
1 & 1 & 1 & ?
\end{array}
\]
Resource Allocation and Fair Division
Problems arise due to the combinatorial structure of the domain of alternatives (eg. allocating indivisible resources), and particular because agents are typically only concerned in resources, not in full allocations. Auctions or negotiation-based approaches well suited, with domain-related constraints (eg. kidney exchanges).

Matching
In double-sided matching problems, each side has preferences on the other side and the objective is to match them. One typically seeks stable states (no pair of agents would be better off leaving their match to form a new pair), eg. stable marriage.
Thanks

Based on joint work, slides, papers, discussions, etc. from/with (in particular):

▶ Yann Chevaleyre
▶ Ulle Endriss
▶ Jérôme Lang
▶ Jérôme Monnot

More on these topics:

▶ International Workshop series “Computational Social Choice”
