Computational aspects of voting

Jérôme Lang
LAMSADE, CNRS & Université Paris-Dauphine

COST-ADT Doctoral School on Computational Social Choice
Estoril, April 9-14, 2010

1. **A very brief introduction to computational social choice**
2. Background
3. Topic 1: computationally hard voting rules
4. Topic 2: voting on combinatorial domains
5. Topic 3: computational aspects of strategyproofness
6. Topic 4: communication issues and incomplete preferences
7. Topic 5: other issues
8. Conclusion

Social choice theory

- most results in social choice theory are of the following form: *there does not exist / there exists a social choice procedure meeting requirements (R1),...,(Rp)*: **impossibility/possibility theorems**.

  Example: Arrow's theorem.

  There exists no aggregation function defined on the set of all profiles, satisfying unanimity, IIA and non-dictatorship.

- *computational issues are neglected*

  Knowing that a given procedure *can* be computed is generally enough.

Computational social choice: two research streams

From social choice theory to computer science

importing concepts and procedures from social choice for solving problems arising in computer science applications, such as

- societies of artificial agents (voting, negotiating / bargaining, ...)
- aggregation procedures for web site ranking and information retrieval
- fair division of computational resources

From computer science to social choice theory

using computational notions and techniques (mainly from AI, OR, Theoretical Computer Science) for solving complex social choice problems.

- computational difficulty of voting rules; exact or approximate algorithms
- voting with a very large (combinatorial) space of alternatives
- computational bareers to manipulation (+ other forms of strategic behaviour)
- communication protocols for voting; voting with incomplete knowledge
- computational aspects of fair division
- several other topics

Outline of the lectures

In the order of appearance:

(Christian Klamler) social choice theory

(José Figueira) history of social choice

(Jérôme Lang) computational social choice: voting

Ulle Endriss computational social choice: fair division

Felix Brandt voting: tournament solutions

Stefano Moretti social choice and game theory: coalitions, power indices

Sébastien Konieczny social choice: logic-based approaches

Thierry Marchant social choice and multicriteria decision analysis

Outline of the lectures

- not enough time to talk about every single piece of work
- for each main topic I'll develop one or two approaches in detail
- focus on computation and communication
- (tentative) full list of references, classified by topic, given in a separate file

1. A very brief introduction to computational social choice
2. **Background**
3. Topic 1: computationally hard voting rules
4. Topic 2: voting on combinatorial domains
5. Topic 3: computational aspects of strategyproofness
6. Topic 4: communication issues and incomplete preferences
7. Topic 5: other issues
8. Conclusion

Voting rules and correspondences

1. a finite *set of voters* $\mathcal{A} = \{1, ..., n\}$;
2. a finite *set of candidates (alternatives)* $\mathcal{X}$;
3. a *profile* = a preference relation (= linear order) on $\mathcal{X}$ for each agent

$$P = (V_1, \ldots, V_n) = (\succ_1, \ldots, \succ_n)$$

   V_i (or $\succ_i$) = *vote* expressed by voter i.
   — there are exceptions, such as in approval voting —

4. $\mathcal{P}^n$ set of all profiles.

Voting rule $F : \mathcal{P}^n \to \mathcal{X}$

$F(V_1, \ldots, V_n)$ = socially preferred (elected) candidate

Voting correspondence $C : \mathcal{P}^n \to 2^{\mathcal{X}} \setminus \{\emptyset\}$

$C(V_1, \ldots, V_n)$ = set of socially preferred candidates.

Rules can be obtained from correspondences by tie-breaking (usually by using a predefined priority order on candidates).

A family of voting rules: *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 $j \Rightarrow score_i(x) = s_j$
- winner: candidate maximizing $s(x) = \sum_{i=1}^{n} score_i(x)$ (+ tie-breaking if necessary)

Examples:

- $s_1 = 1, s_2 = \ldots = s_p = 0 \Rightarrow$ *plurality*;
- $s_1 = s_2 = \ldots = s_{p-1} = 1, s_p = 0 \Rightarrow$ *veto* ;
- $s_1 = p-1, s_2 = p-2, \ldots s_p = 0 \Rightarrow$ *Borda*.

2 voters	1 voter	1 voter
c	a	d
b	b	a
a	d	b
d	c	c

plurality	Borda	veto
$a \mapsto 1$	$a \mapsto 6$	$a \mapsto 6$
$b \mapsto 0$	$b \mapsto 7$	$b \mapsto 6$
$c \mapsto 2$	$c \mapsto 6$	$c \mapsto 4$
$d \mapsto 1$	$d \mapsto 4$	$d \mapsto 4$
c winner	b winner	a ou b winner

Condorcet winner

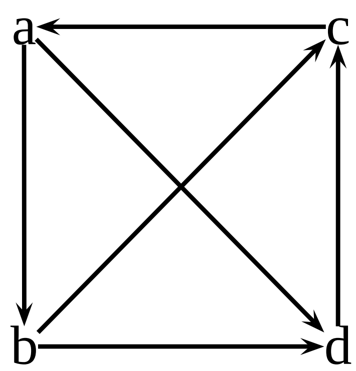
$N(x,y) = \#\{i, x \succ_i y\}$ number of voters who prefer x to y.

Condorcet winner: 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).

a	d	c
b	b	a
d	c	b
c	a	d

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: $d \succ c$

majority graph:



No Condorcet winner.

Condorcet winner

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

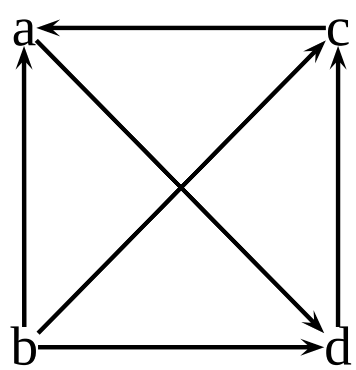
Condorcet winner: a candidate x such that $\forall y \neq x$, $N(x,y) > \frac{n}{2}$
(= a candidate who beats any other candidate by a majority of votes).

A *Condorcet-consistent rule* elects the Condorcet winner whenever there is one.

a	d	c
b	b	$\mathbf{b}$
d	c	$\mathbf{a}$
c	a	d

2 voters out of 3: $\mathbf{b} \succ \mathbf{a}$
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: $d \succ c$

majority graph :



b Condorcet winner.

Another family of voting rules: *Condorcet-consistent rules*

A *Condorcet-consistent rule* elects the Condorcet winner whenever there is one.

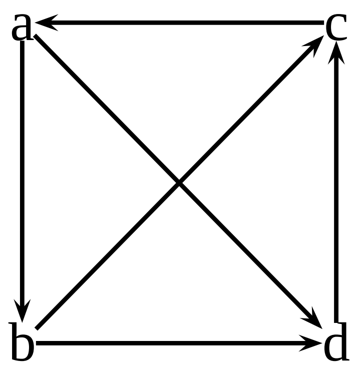
An example: the *Copeland rule*:

$C(x)$ = number of candidates y such that a majority of voters prefers x to y.

Copeland winner = candidate maximizing C.

a	d	c
b	b	a
d	c	b
c	a	d

majority graph :



$C(a) = 2$
$C(b) = 2$
$C(c) = 1$
$C(d) = 1$
a and b pre-winners
(the winner is obtained after tie-breaking)

Important note: no scoring is Condorcet-consistent.

***Simple transferable vote* (STV)**

if there exists a candidate *c* ranked first by more than 50% of the 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 more than 50% of the votes

When there are only 3 candidates, STV coincides with *plurality with runoff*.

3	4	3	2
a	*b*	*c*	*d*
d	*d*	*d*	*c*
b	*a*	*a*	*b*
c	*c*	*b*	*a*

3	4	3	2
a	*b*	*c*	*c*
b	*a*	*a*	*b*
c	*c*	*b*	*a*

7	5
b	*c*
c	*b*

Winner: *b*

1. Introduction to computational voting theory
2. Background
3. **Topic 1: computationally hard voting rules**
4. Topic 2: voting on combinatorial domains
5. Topic 3: computational aspects of strategyproofness
6. Topic 4: communication issues and incomplete preferences
7. Topic 5: other issues
8. Conclusion

A brief refresher on computational complexity

A **decision problem** is a pair $P = \langle I_P, Y_P \rangle$ where

- I_P set of *problem instances*
- Y_P set of *positive instances*

$N_P = I_P \setminus Y_P$ set of *negative instances*

A decision problem is usually identified with the language Y_P of positive instances.

Algorithm for a decision problem:

A decision problem P is solved by an algorithm A if A halts for every instance $x \in I_P$, and returns YES if and only if $x \in Y_P$. We also say that the set (or the language) Y_P is recognized by A.

A **search problem** is a triple $P = \langle I_P, S_P, R \rangle$ where

- I_P set of *problem instances*
- S_P set of *positive solutions*
- $R \subseteq I_P \times S_P$ [$R(x,s)$ means that s is a solution for x]

Complexity classes for decision problems

Let A be an algorithm running on a set of instances I. Let $x \in I$.

- $\hat{t}_A(x)$ = running time of A on x ($\approx$ number of elementary steps);
- the worst-case running time of A is the function $t_A : \mathbb{N} \rightarrow \mathbb{N}$ defined by

$$t_A(n) = max\{\hat{t}_A(x) | x \in I, |x| \leq n\}$$

- the running time of A is in $O(g(n))$ if $t_A(n)$ is $O(g(n))$.

[[given two functions $f, g : \mathbb{N} \rightarrow \mathbb{N}$, we say that $f(n)$ is $O(g(n))$ if there exist constants c, a and n_0 such that for all $n \geq n_0$, $f(n) \leq c.g(n) + a$.]]

A decision problem can be solved with time $f(n)$ if there exist an algorithm A that solves it and whose running time (resp. space) is in $O(f(n))$.

Deterministic polynomial time:

P = set of all decision problems that can be solved in time n^k for some $k \in \mathbb{N}$

Nondeterministic algorithm: apart from all usual constructs, can execute commands of the type guess $y \in \{0,1\}$.

Structure of a nondeterministic algorithm = computation tree (guess instructions corresponding to branching points) $\neq$ linear structure of a deterministic algorithm (at any step, one possible next step).

Nondeterministic problem solution:

$P = \langle I_P, Y_P \rangle$ decision problem.

A nondeterministic algorithm A solves P if, for all $x \in I_P$:

1. A running on x halts for any possible guess sequence;
2. $x \in Y_P$ iff there exists a sequence of guesses which leads A to return the value YES.

Nondeterministic polynomial time:

NP = set of all decision problems that can be solved by a nondeterministic algorithm in time n^k for some $k \in \mathbb{N}$

Equivalently, NP is the set of all decision problems for which *a solution can be verified in deterministic polynomial time*.

NP-complete problems

A decision problem is NP-hard if any problem of NP can be polynomially reduced to it.

A decision problem is NP-complete if it is in NP and it is NP-hard.

NP-complete problems = "the hardest" among problems in NP

coNP = set of all decision problems whose complement in in NP

$P = \langle I_P, Y_P \rangle$; Y_P is in coNP if and only if $I_P \setminus I_P$ is in NP.

Oracles and relativized complexity classes

Oracles: let $P = \langle I_P, S_P, R\rangle$ be a function problem. An *oracle for P* is an abstract device which, for any $x \in IP$, returns a value $f(x) \in SP$ in just one computation step.

An NP-oracle is an abstract device which, for any x ? IP , returns a value f (x) ? SP in just one computation step.

A decision problem

C complexity class

C^{NP} is the class of all decision problems that can be recognized with complexity C by an algorithm using oracles for a NP-complete problem P.

The (second level of the) polynomial hierarchy

- $\Theta_2^{\mathsf{P}} = \Delta_2^{\mathsf{P}}(O(\log n))$ = set of all decision problems that can be solved in deterministic polynomial time using a logarithmic number of NP-oracles.
- $\Sigma_2^p = \mathsf{NP}^{\mathsf{NP}}$ = set of all decision problems that can be solved in nondeterministic polynomial time using NP-oracles.
- $\Pi_2^p = \mathrm{co}\Sigma_2^p$

Computing voting rules

Most voting rules can be computed in polynomial time

Examples:

- positional scoring rules, plurality with runoff: $O(np)$
- Copeland, maximin, STV: $O(np^2)$

But some voting rules are NP-hard.

Hard rules: a classification

- rules based on the majority graph: *tournament solutions* (among which Slater, Banks, Tournament Equilibrium Set)
  ⇒ lecture by Felix Brandt
- rules based on the weighted majority graph: Kemeny
- other rules: Dodgson, Young

Hard rules: Kemeny

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

- *Kemeny distance*:

$$d_K(V,V') = \text{number of } (x,y) \in \mathcal{X}^2 \text{ on which } V \text{ and } V' \text{ disagree}$$

$$d_K(V,\langle V_1,\ldots,V_n\rangle) = \sum_{i=1,\ldots,n} d_K(V,V_i)$$

- *Kemeny consensus* = linear order $\succ^*$ such that $d_K(\succ^*,\langle V_1,\ldots,V_n\rangle)$ minimum
- *Kemeny winner* = candidate ranked first in a Kemeny consensus

Hard rules: Kemeny

A characterization of Kemeny: with each profile $P = \langle P_1, \ldots, P_n \rangle$ associate the pairwise comparison matrix $(N(x,y))_{x,y \in X}$ where $N(x,y)$ is the number of voters who prefer x to y.

Given a ranking R:

$$K(R) = \sum_{(x,y) \in R} N_(x,y)$$

If $x > y$ is in R then this corresponds to $N(x,y)$ agreements (and $N(y,x)$ disagreements)

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

4 voters	3 voters	2 voters
a	b	c
b	c	a
c	a	b

Find the Kemeny winner(s).

Hard rules: Kemeny

4 voters	3 voters	2 voters
a	*b*	*c*
b	*c*	*a*
c	*a*	*b*

N	*a*	*b*	*c*
a	–	6	4
b	3	–	7
c	5	2	–

Kemeny scores:

abc	*acb*	*bac*	*bca*	*cab*	*cba*
17	12	14	15	13	10

Kemeny consensus: *abc*; Kemeny winner: *a*

Hard rules: Kemeny

- early results: Kemeny is NP-hard (Orlin, 81; Bartholdi *et al.*, 89; Hudry, 89)
- deciding whether a candidate is a Kemeny winner is $\Delta_2^{\mathsf{P}}(O(\log n))$-complete (Hemaspaandra, Spakowski & Vogel, 04): needs logarithmically many oracles.

Membership to $\Delta_2^{\mathsf{P}}(O(\log n))$:

1. $kmin := 0; kmax := \frac{nm(m-1)}{2}$;
2. **Repeat**
3. $\quad k := \lceil \frac{kmin+kmax}{2} \rceil$;
4. $\quad$ **if** there exists a ranking R such that $K(R) \geq k$
5. $\quad$ **then** $kmax := k$
6. $\quad$ **else** $kmin := k-1$
7. **Until** $kmin = kmax$
8. $k^* := kmin (= kmax)$
9. guess a ranking R;
10. check that $K(R) = k^*$ and that $top(R) = x$.

Step 4: NP-oracle [4a. guess R; 4b. check that $K(R) \geq k$]

Hard rules: Kemeny

Lots of other works on Kemeny, among which

- efficient computation: Davenport and Kalagnanam, 04; Conitzer, Davenport and Kalagnanam, 06; Betzler, Fellows, Guo, Niedermeier & Rosamond, 09.
- fixed-parameter complexity: Betzler, Fellows, Guo, Niedermeier & Rosamond, 08.
- approximation: Ailon, Charikar & Newman, 05; Kenyon-Mathieu and Schudy, 07.

More general problem: *median orders* (survey in Hudry (08).

Hard rules: Dodgson

For any $x \in \mathcal{X}$, $D(x)$ = smallest number of elementary changes needed to make x a Condorcet winner.

elementary change = exchange of adjacent candidates in a voter's ranking

Dodgson winner(s): candidate(s) minimizing $D(x)$

An example (Nurmi, 04):

10 voters	8 voters	7 voters	4 voters
c	*d*	*d*	*b*
b	*a*	*b*	*a*
a	*b*	*a*	*c*
d	*c*	*c*	*d*

Find the Dodgson winner.

Hard rules: Dodgson

For any $x \in \mathcal{X}$, $D(x)$ = smallest number of elementary changes needed to make x a Condorcet winner.

elementary change = exchange of adjacent candidates in a voter's ranking

Dodgson winner(s): candidate(s) minimizing $D(x)$

An example (Nurmi, 04):

10 voters	8 voters	7 voters	4 voters
c	d	d	b
b	a	b	a
a	b	a	c
d	c	c	d

Dodgson winner: D, although D is the Condorcet loser.

Who is the winner if all votes are reversed?

Hard rules: Dodgson

Another example (Brandt, 09):

2	2	2	2	2	1	1
d	*b*	*c*	*d*	*a*	*a*	*d*
c	*c*	*a*	*b*	*b*	*d*	*a*
a	*a*	*b*	*c*	*c*	*b*	*b*
b	*d*	*d*	*a*	*d*	*c*	*c*

Replace every voter by three voters:

6	6	6	6	6	3	3
d	*b*	*c*	*d*	*a*	*a*	*d*
c	*c*	*a*	*b*	*b*	*d*	*a*
a	*a*	*b*	*c*	*c*	*b*	*b*
b	*d*	*d*	*a*	*d*	*c*	*c*

Hard rules: Dodgson

Another example (Brandt, 09): Dodgson does not satisfy *homogeneity*

2	2	2	2	2	1	1
d	*b*	*c*	*d*	*a*	*a*	*d*
c	*c*	*a*	*b*	*b*	*d*	*a*
a	*a*	*b*	*c*	*c*	*b*	*b*
b	*d*	*d*	*a*	*d*	*c*	*c*

Dodgson winner: *A*

Replace every voter by three voters:

6	6	6	6	6	3	3
d	*b*	*c*	*d*	*a*	*a*	*d*
c	*c*	*a*	*b*	*b*	*d*	*a*
a	*a*	*b*	*c*	*c*	*b*	*b*
b	*d*	*d*	*a*	*d*	*c*	*c*

Dodgson winner: *D*

Hard rules: Dodgson

Although Dodgson has received much attention in the last years, it fails to satisfy many desirable properties (Brandt, 09): Smith consistency, homogeneity, monotonicity, independence of clones.

Moreover, computing Dodgson is hard:

- Bartholdi, Tovey & Trick, 89: deciding whether *x* is a Dodgson winner is NP-hard.
- Hemaspaandra, Hemaspaandra & Rothe, 97: deciding whether *x* is a Dodgson winner is Θ_2^{P}-complete (= requires a logarithmic number of calls to NP oracles)

Caragiannis, Kaklamanis, Karanikolas & Procaccia (10): *socially desirable approximations of Dodgson*. Example: *monotonic approximations* = voting rules:

- satisfying monotonicity
- close enough to Dodgson
- (possibly) computable in polynomial time

The approximation of a voting rule is a new voting rule that may be interesting *per se*!

Hard rules: Dodgson

For all candidates $x, y \neq x$: $Deficit(x,y) = \max(0, 1 + \lfloor \frac{N(y,x)-N(x,y)}{2} \rfloor)$
($Deficit(x,y)$ = number of votes (if any) that x needs to gain in order to beat y)

Tideman score:

$$T(x) = \sum_{y \neq x} Deficit(x,y)$$

Tideman winner(s) = candidate(s) with the lowest Tideman score

- Tideman winners are computable in time $O(n.p^2)$
- Tideman satisfies monotonicity and homogeneity
- (after some rescaling of the definition of the Tideman score) Tideman is an approximation of Dodgson with approximation ratio $O(m.\log m)$: $T(x) \leq \rho.D(x)$ with $\rho = O(m.\log m)$ (Caragiannis, Kaklamanis, Karanikolas & Procaccia, 10)
- under the impartial culture assumption (uniform distribution of profiles), the probability that the Tideman winner and the Dodgson winner coincide converges asymptotically to 1 as the number of voters tends to infinity (McCabe-Dansted, Pritchard and Slinko, 06)

Hard rules: Dodgson

Recall that $Deficit(x,y) = \max(0, 1 + \lfloor \frac{N(y,x) - N(x,y)}{2} \rfloor)$ = number of votes (if any) that x needs to gain in order to beat y by a majority of votes.

Define $Swap(x,y)$ = number of votes in which y is immediately above x.

- if for every $y \neq x$, $Swap(x,y) \geq Deficit(x,y)$ then the Dodgson score of x is $\sum_{y \neq x} Swap(x,y)$.
- therefore, if $Swap(x,y) \geq Deficit(x,y)$ holds for every x, y, then the Dodgson winner can be computed in polynomial time.
- under the impartial culture assumption, the probability that $Swap(x,y) \geq Deficit(x,y)$ holds for every x, y tends to 1 when the number of voters n tends to infinity (Homan and Hemaspaandra, 06).

Hard rules: Young

For any $x \in \mathcal{X}$, $Y(x)$ = smallest number of elementary changes needed to make x a Condorcet winner.

elementary change = removal of a voter

4 voters	2 voters	3 voters
a	b	c
b	c	e
c	e	d
d	d	a
e	a	b

Find the Young winner(s).

Deciding whether x is a Young winner is Θ_2^{P}-complete (Rothe, Spakowski & Vogel, 03)

Hard rules: Banks

- M_P majority graph induced by P;
- x is a Banks winner if x is undominated in some maximal transitive subset of M_P.
- deciding whether x is a Banks winner is NP-complete (Woeginger, 2003; Brandt *et al.*, 2009)
- however, it is possible to find an arbitrary Banks winner in polynomial time (Hudry, 2004)

Finding a Banks winner in polynomial time by a greedy algorithm:

> $A := \{x\}$ where x is an arbitrary candidate;
> **repeat**
>   find y such that the subgraph of M_P restricted to $A \cup \{y\}$ is transitive;
>   add y to A
> **until** there is no such y;
> **return** c undominated in A

Hard rules: Banks

4 voters	2 voters	3 voters
a	*b*	*c*
b	*c*	*e*
c	*e*	*d*
d	*d*	*a*
e	*a*	*b*

Find the Banks winner(s).

Hard rules: Slater

$P = (V_1, \dots, V_n)$ profile

- M_P majority graph induced by P.
- distance of a linear order V to M_P: number of edges in M_P disagreeing with V.
- Slater ranking = linear order on $\mathcal{X}$ minimising the distance to M_P.
- Slater winner: best candidate in some Slater ranking

Complexity:

- weak tournaments (with possible ties): Θ_p^2-complete;
- tournaments: NP-hard, in Θ_p^2

Hard rules: Slater

4 voters	2 voters	3 voters
a	*b*	*c*
b	*c*	*e*
c	*e*	*d*
d	*d*	*a*
e	*a*	*b*

Find the Slater winner(s).

1. Introduction to computational social choice
2. Background
3. Topic 1: computationally hard voting rules
4. **Topic 2: voting on combinatorial domains**
5. Topic 3: computational aspects of strategyproofness
6. Topic 4: communication and incomplete knowledge
7. Topic 5: other issues

Key question: *structure* of the set $\mathcal{X}$ of candidates?

Example 1 choosing a common menu:

$$\begin{aligned} \mathcal{X} = \quad & \{\texttt{asparagus risotto, foie gras}\} \\ \times \quad & \{\texttt{roasted chicken, vegetable curry}\} \\ \times \quad & \{\texttt{white wine, red wine}\} \end{aligned}$$

Example 2 multiple referendum: a local community has to decide on several interrelated issues (should we build a swimming pool or not? should we build a tennis court or not?)

Example 3 choosing a joint plan: the group travel problem (Klamler & Pfirschy). A set of cities; a set of agents; each of whom has preferences over edges between cities. The group will travel together and has to reach every city once.

Example 4 recruiting committee (3 positions, 6 candidates):

$\mathcal{X} = \{A \mid A \subseteq \{a,b,c,d,e,f\}, \ |A| \leq 3\}$.

Combinatorial domains: $\mathcal{V} = \{X_1, \ldots, X_p\}$ set of *variables*, or *issues*;
$\mathcal{X} = D_1 \times \ldots \times D_p$ (where D_i is a finite value domain for variable X_i)

How should such a vote be conducted?

Some classes of solutions:

1. vote separately on each variable, in parallel.
2. ask voters to specify their preference relation by ranking all alternatives *explicitly*.
3. limit the number of possible alternatives that voters may vote for.
4. ask voters to report only a small part of their preference relation and appply a voting rule that needs this information only, such as plurality.
5. ask voters their preferred alternative(s) and complete them automatically using a predefined *distance*.
6. *sequential voting* : decide on every variable one after the other, and broadcast the outcome for every variable before eliciting the votes on the next variable.
7. use a *compact preference representation language* in which the voters' preferences are represented in a concise way.

How should such a vote be conducted?

Some classes of solutions:

1. **don’t bother and vote separately on each variable.**

How should such a vote be conducted?

Some classes of solutions:

1. **don't bother and vote separately on each variable.**: *multiple election paradoxes* arise as soon as some voters have preferential dependencies between attributes.

Example

2 binary variables S (build a new swimming pool), T (build a new tennis court)

voters 1 and 2	$S\bar{T} \succ \bar{S}T \succ \bar{S}\bar{T} \succ ST$
voters 3 and 4	$\bar{S}T \succ S\bar{T} \succ \bar{S}\bar{T} \succ ST$
voter 5	$ST \succ S\bar{T} \succ \bar{S}T \succ \bar{S}\bar{T}$

Problem 1: voters 1-4 feel ill at ease reporting a preference on $\{S,\bar{S}\}$ and $\{T,\bar{T}\}$

Problem 2: suppose they do so by an "optimistic" projection

- voters 1, 2 and 5: S; voters 3 and 4: $\bar{S} \Rightarrow$ decision = S;
- voters 3,4 and 5: T; voters 1 and 2: $\bar{T} \Rightarrow$ decision = T.

Alternative ST is chosen although it is the worst alternative for all but one voter.

How should such a vote be conducted?

Some classes of solutions:

1. **don't bother and vote separately on each variable.** *multiple election paradoxes* arise as soon as some voters have preferential dependencies between attributes.

Not too bad when preferences are *separable*: *the preference over the possibles values of a variable is independent from the values of other variables*

Separability:

$\mathcal{V} = \{X_1, \ldots, X_p\}$ set of variables

$\mathcal{X} = D_1 \times ... \times D_p$

$D_{-i} = \times_{j \neq i} D_j$

$$\text{for every } X_i \in \mathcal{V}, \text{ every } \vec{x}_{-i}, \vec{x}'_{-i} \in D_{-i}, \text{ and every } x_i, x'_i \in D_i,$$
$$(\vec{x}_{-i}, x_i) \succeq (\vec{x}_{-i}, x'_i) \text{ if and only if } (\vec{x}'_{-i}, x_i) \succeq (\vec{x}'_{-i}, x'_i)$$

x_i is preferred to x'_i for some tuple of values $\vec{x}_{-i}$ of the other variables *iff* x_i is preferred to x'_i for any other tuple of values $\vec{x}'_{-i}$ of the other variables.

How should such a vote be conducted?

Some classes of solutions:

1. don't bother and vote separately on each variable.
2. **ask voters to specify their preference relation by ranking all alternatives *explicitly***.

$\mathcal{V} = \{X_1, \ldots, X_p\}$; $\mathcal{X} = D_1 \times \ldots \times D_p$

There are $\Pi_{1 \leq i \leq p} |D_i|$ alternatives.

$\Rightarrow$ **as soon as there are more than three or four variables, explicit preference elicitation is irrealistic.**

How should such a vote be conducted?

Some classes of solutions:

1. vote separately on each variable, in parallel.
2. ask voters to specify their preference relation by ranking all alternatives *explicitly*.
3. **limit the number of possible alternatives that voters may vote for.**

- *arbitrary* (who decides which alternatives are allowed?)
- so that this solution be realistic, the number of alternatives the voters can vote for has to be low. Therefore, voters only express their preferences on a tiny fraction of the alternatives.

How should such a vote be conducted?

Some classes of solutions:

1. don't bother and vote separately on each variable.
2. ask voters to specify their preference relation by ranking all alternatives *explicitly*.
3. limit the number of possible alternatives that voters may vote for.
4. **ask voters to report only a small part of their preference relation and appply a voting rule that needs this information only, such as plurality.**

Results are completely nonsignificant as soon as the number of variables is much higher than the number of voters ($2^p \gg n$).

5 voters, 2^6 alternatives; rule : plurality

001010: 1 vote; 010111: 1 vote; 011000: 1 vote; 101001: 1 vote; 111000: 1 vote
all other candidates : 0 vote.

How should such a vote be conducted?

Some classes of solutions:

1. don't bother and vote separately on each variable.
2. ask voters to specify their preference relation by ranking all alternatives *explicitly*.
3. limit the number of possible alternatives that voters may vote for.
4. ask voters to report only a small part of their preference relation and appply a voting rule that needs this information only, such as plurality.
5. **ask voters their preferred alternative(s) and complete them automatically using a predefined *distance*.**

5 **ask voters their preferred alternative(s) and complete them automatically using a predefined *distance*.**

- every voter specifies one preferred alternatives $\vec{x}^*$;
- for all alternatives $\vec{x}, \vec{y} \in D$, $\vec{x} \succ_i \vec{y}$ if and only if $d(\vec{x}, \vec{x}^*) < d(\vec{y}, \vec{x}^*)$, where d is a predefined distance on D.

$+$ cheap in elicitation an computation.

$-$ important domain restriction.

Two examples of such approaches:

- propositional merging (Konieczny & Pino-Perez 98, etc.)
- minimax approval voting

Minimax approval voting (Brams, Kilgour & Sanver, 2007)

- n voters, m candidates, $k \leq m$ positions to be filled
- each voter casts an approval ballot $V_i = (v_i^1, \ldots, v_i^m) \in \{0,1\}^m$
  $v_i^j = 1$ if voter i approves candidate j.
- for every subset Y of k candidates,
  - $d(Y, V_i)$ = Hamming distance between Y and V_i (number of disagreements)
  - $d(Y, (V_1, \ldots, V_n)) = \max_{i=1,\ldots,n} d(Y, V_i)$
  - find Y minimizing $d(Y, (V_1, \ldots, V_n))$

Example: $n = 4$, $m = 4$, $k = 2$.

	1110	1101	1010	1010	*sum*	*max*
1100	1	1	2	2	6	**2**
1010	1	3	0	0	**4**	3
1001	3	1	3	3	10	3
0110	1	3	2	2	8	3
0101	3	1	4	4	12	4
0011	3	3	2	2	10	3

Minimax approval voting

- finding an optimal subset is NP-hard (Frances & Litman, 97)
- (Le Grand, Markakis & Mehta, 07): approximation algorithms for minimax approval voting

1. pick arbitrarily one of the ballots V_j
2. $k_j \leftarrow$ number of 1's in V_j
3. **if** $k_j > k$ **then** pick $k_j - k$ coordinates in V_j and set them to 0;
4. **if** $k_j < k$ **then** pick $k - k_j$ coordinates in V_j and set them to 1;
5. **return** the modified ballot V_j'

The above algorithm is a polynomial 3-approximation of minimax approval (Le Grand, Markakis & Mehta, 07)

Minimax approval voting

The above algorithm is a polynomial 3-approximation of minimax approval (Le Grand, Markakis & Mehta, 07)

- let V^* be a minimax committee and $OPT = d(V^*, (V_1, \ldots, V_n))$.
- let V_j the ballot picked by the algorithm.
- $d(V'_j, V_i) \leq d(V'_j, V_j) + d(V_j, V^*) + d(V^*, V_i)$;
- $d(V^*, V_i) \leq OPT$ and $d(V_j, V^*) \leq OPT$;
- by construction of V'_j, $d(V'_j, V_j) \leq d(V^*, V_j) \leq OPT$;
- therefore $d(V', V_i) \leq 3OPT$

Conclusion: $d(V', (V_1, \ldots, V_n)) \leq 3OPT$.

Better approximation (ratio 2) in (Caragiannis, Kalaitzis & Markakis, 10)

More generally: multiwinner elections (Meir, Procaccia, Rosenschein & Zohar, 08)

How should such a vote be conducted?

Some classes of solutions:

1. don't bother and vote separately on each variable.
2. ask voters to specify their preference relation by ranking all alternatives *explicitly*.
3. limit the number of possible alternatives that voters may vote for.
4. ask voters to report only a small part of their preference relation and appply a voting rule that needs this information only, such as plurality.
5. ask voters their preferred alternative(s) and complete them automatically using a predefined *distance*.
6. ***sequential voting*** **: decide on every variable one after the other, and broadcast the outcome for every variable before eliciting the votes on the next variable.**

Sequential voting

voters 1 and 2 $\quad S\bar{T} \succ \bar{S}T \succ \bar{S}\bar{T} \succ ST$

voters 3 and 4 $\quad \bar{S}T \succ S\bar{T} \succ \bar{S}\bar{T} \succ ST$

voter 5 $\quad ST \succ S\bar{T} \succ \bar{S}T \succ \bar{S}\bar{T}$

Fix the order $S > T$.

Step 1 elicit preferences on $\{S, \bar{S}\}$

if voters report preferences optimistically: $3 : S \succ \bar{S}$; $2 : \bar{S} \succ S$

Step 2 compute the local outcome and broadcast the result

S

Step 3 elicit preferences on $\{T, \bar{T}\}$ given the outcome on $\{S, \bar{S}\}$

4: $S : \bar{T} \succ T$; 1: $S : T \succ \bar{T}$

Step 4 compute the final outcome

$S\bar{T}$

Sequential voting

+ simple elicitation protocol
+ computationally easy (provided local rules are easy to compute)

- restriction-free sequential voting
  - + always applicable
  - – voters may feel ill at ease reporting a preference on some attributes, or experience regret after the final outcome is known
  - – the outcome depends on the order in which the attributes are decided
- “safe” sequential voting

voters 1 and 2	$S\bar{T} \succ \bar{S}T \succ \bar{S}\bar{T} \succ ST$
voters 3 and 4	$\bar{S}T \succ S\bar{T} \succ \bar{S}\bar{T} \succ ST$
voter 5	$ST \succ S\bar{T} \succ \bar{S}T \succ \bar{S}\bar{T}$

Suppose voters behave optimistically, and that the chair knows that.

$\mathbf{S} > \mathbf{T}$

3 votes for S, 2 votes for $\bar{S}$; *local outcome*: S
given $\mathbf{S} = S$, 4 votes for $\bar{T}$, 1 vote for $T \Rightarrow \bar{T}$; *final outcome*: $S\bar{T}$

$\mathbf{T} > \mathbf{S}$

3 votes for T, 2 votes for $\bar{T}$; *local outcome*: T
given $\mathbf{T} = T$, 4 votes for $\bar{S}$, 1 vote for $S \Rightarrow \bar{S}$; *final outcome*: $\bar{S}T$

The chair's strategy:

- if she prefers $S\bar{T}$ to $\bar{S}T$: choose the order $\mathbf{S} > \mathbf{T}$
- if she prefers $\bar{S}T$ to $S\bar{T}$: choose the order $\mathbf{T} > \mathbf{S}$

Note that ST and $\bar{S}\bar{T}$ cannot be obtained.

The chair can (sometimes) control the election by fixing the agenda

“Safe” sequential voting

$O : X_1 > \ldots > X_p$ order on variables

At step i, all voters vote on variable X_i, using a local voting rule r_i, and the outcome is communicated to the voters before variable X_{i+1} is considered.

Requires the domain restriction

(R) *the preferences of every voter on X_i are independent from the values of $X_{i+1}, \ldots, X_n$.*

+ simple elicitation protocol
+ computationally easy (provided local rules are easy to compute)
+ voters have no problem reporting their preferences, nor do they ever experience regret after the final outcome is known
– the number of profiles satisfying (R) is exponentially small; however
  + many “practical” domains comply with (R)

    `main course` > `first course` > `wine`

  + still: much weaker restriction than separability.

“Safe” sequential voting

Conditional preferential independence (Keeney & Raiffa, 76)

$\{\mathcal{X}, \mathcal{Y}, \mathcal{Z}\}$ partition of $\mathcal{V}$.

$D_{\mathcal{X}} = \times_{X_i \in \mathcal{X}} D_i$ etc.

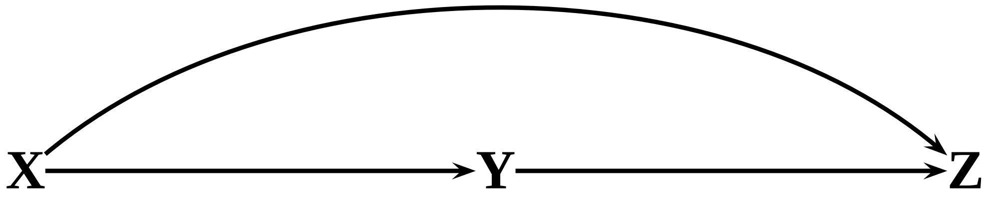
$\mathcal{X}$ is preferentially independent of $\mathcal{Y}$ (given $\mathcal{Z}$) iff

$$\text{for all } x, x' \in Dom(\mathcal{X}), v, v' \in Dom(\mathcal{Y}), w \in Dom(\mathcal{Z}),$$
$$(x,y,z) \succeq (x',y,z) \text{ if and only if } (x,y',z) \succeq (x',y',z)$$

given a fixed value z of Z, the preferences over the possibles values of X is independent from the value of Y

CP-nets (Boutilier, Brafman, Hoos and Poole, 99)

Language for specifying preferences on combinatorial domains based on the notion of conditional preferential independence.



$x \succ \bar{x}$

$x : y \succ \bar{y}$
$\bar{x} : \bar{y} \succ y$

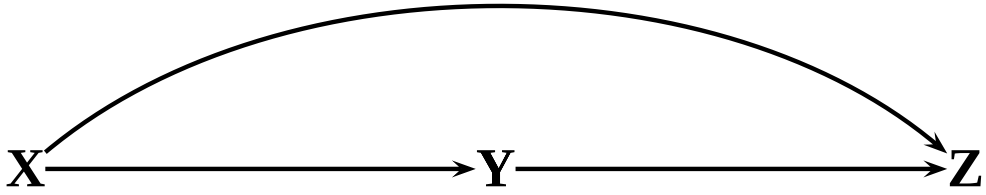
$x \vee y : z \succ \bar{z}$
$\neg(x \vee y) : \bar{z} \succ z$

X independent of **Y** and **Z**; **Y** independent of **Z**

$x : y \succ \bar{y}$ | if $X = x$
then $Y = y$ preferred to $Y = \bar{y}$
everything else (z) being equal (*ceteris paribus*)

$$xyz \succ x\bar{y}z; \quad xy\bar{z} \succ x\overline{yz};$$
$$\overline{xy}z \succ \bar{x}yz; \quad \overline{xyz} \succ \bar{x}y\bar{z}$$

CP-nets



$x \succ \bar{x}$	$x : y \succ \bar{y}$ $\bar{x} : \bar{y} \succ y$	$x \vee y : z \succ \bar{z}$ $\neg(x \vee y) : \bar{z} \succ z$

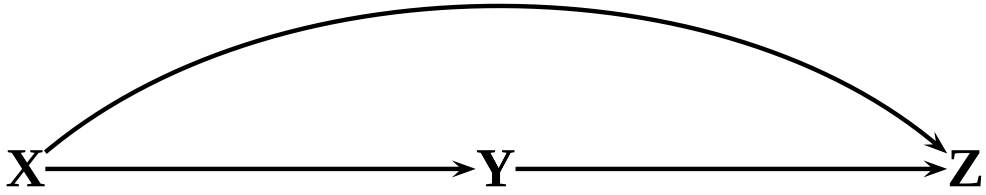
$\succ^X$: $xyz \succ \bar{x}yz$, $xy\bar{z} \succ \bar{x}y\bar{z}$, $x\bar{y}z \succ \overline{xy}z$, $x\overline{yz} \succ \overline{xyz}$

$\succ^Y$: $xyz \succ x\bar{y}z$, $xy\bar{z} \succ x\overline{yz}$, $\overline{xy}z \succ \bar{x}yz$, $\overline{xyz} \succ \bar{x}y\bar{z}$

$\succ^Z$: $xyz \succ xy\bar{z}$, $x\bar{y}z \succ x\overline{yz}$, $\bar{x}yz \succ \overline{xy}z$, $\overline{xyz} \succ \overline{xy}z$

$\succ_{\mathcal{C}}$ = transitive closure of $\succ^X \cup \succ^Y \cup \succ^Z$

CP-nets

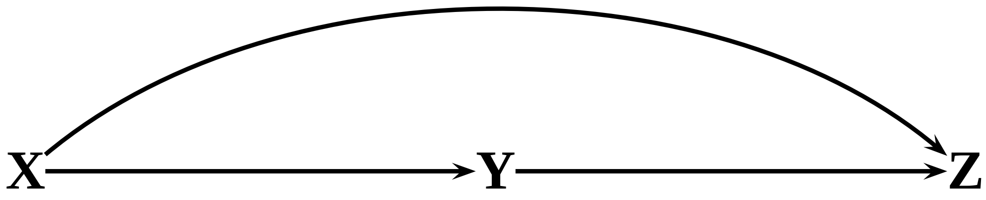


$x \succ \bar{x}$

$x : y \succ \bar{y}$
$\bar{x} : \bar{y} \succ y$

$x \vee y : z \succ \bar{z}$
$\neg(x \vee y) : \bar{z} \succ z$

$$\succ: \quad xyz \;\begin{matrix} \nearrow & x\bar{y}z & \searrow \\ \searrow & xy\bar{z} & \nearrow \end{matrix}\; x\overline{yz} \longrightarrow \overline{xyz} \longrightarrow \overline{xy}z \longrightarrow \bar{x}yz \longrightarrow \bar{x}y\bar{z}$$



1. elicit voters' preferences on **X** (possible because their preferences on **X** are unconditional);
2. apply local voting rule r_X and determine the "local" winner x^*;
3. elicit voters' preferences on **Y** given **X** = x^* (possible because their preferences on **Y** depend only on **X**);
4. apply local voting rule r_Y and determine y^*;
5. elicit voters' preferences on **Z** given **X** = x^* and **Y** = y^*.
6. apply local voting rule r_Z and determine z^*.
7. winner: (x^*, y^*, z^*)

Example: $r_X = r_Y =$ majority rule

3 voters	2 voters	2 voters
$\bar{x}y \succ \overline{xy} \succ x\bar{y} \succ xy$	$xy \succ x\bar{y} \succ \overline{xy} \succ \bar{x}y$	$x\bar{y} \succ xy \succ \bar{x}y \succ \overline{xy}$

For all voters, X is preferentially independent of Y: $\mathcal{G} = \{(X,Y)\}$

$\succ^X$:

3 voters	2 voters	2 voters
$\bar{x} \succ x$	$x \succ \bar{x}$	$x \succ \bar{x}$

4 voters unconditionally prefer x over $\bar{x} \Rightarrow x^* = r_X(\succ_1, \ldots, \succ_7) = x$

Example: $r_X = r_Y =$ majority rule

3 voters	2 voters	2 voters
$\bar{x}y \succ \bar{x}\bar{y} \succ x\bar{y} \succ xy$	$xy \succ x\bar{y} \succ \bar{x}\bar{y} \succ \bar{x}y$	$x\bar{y} \succ xy \succ \bar{x}y \succ \bar{x}\bar{y}$

$$x^* = r_X(\succ_1, \ldots, \succ_7) = x$$

$\succ^{Y|X=x}$:

3 voters	2 voters	2 voters
$\bar{y} \succ y$	$y \succ \bar{y}$	$\bar{y} \succ y$

given $X = x$, 5 voters out of 7 prefer $\bar{y}$ to $y \Rightarrow y^* = r^{Y|X=x}(\succ_1, \ldots, \succ_7) = \bar{y}$

$$Seq(r_X, r_Y)(\succ_1, \ldots, \succ_7) = (x, \bar{y})$$

Question: given some property P of voting rules, do we have

$$r_1, \ldots, r_p \text{ satisfy } P \Rightarrow Seq(r_1, \ldots, r_p) \text{ satisfies } P?$$

General study in (Lang & Xia, 09); here we just give an example for *participation*

Counter-example for *participation*

two variables X, Y. $D_X = \{x_0, x_1, x_2\}$; $D_Y = \{y, \overline{y}\}$.

r_1 a scoring rule with score vector $(3,2,0)$, r_2 = majority.

r_1 and r_2 satisfy participation.

V_1, V_2: $x_0 y \succ x_0 \overline{y} \succ x_1 \overline{y} \succ x_1 y \succ x_2 \overline{y} \succ x_2 y$

$$\boxed{x_0 \succ x_1 \succ x_2} \qquad \boxed{\begin{array}{l} x_0 : y \succ \overline{y} \\ x_1 : \overline{y} \succ y \\ x_2 : \overline{y} \succ y \end{array}}$$

V_3: $x_1 y \succ x_2 y \succ x_0 y \succ x_1 \overline{y} \succ x_2 \overline{y} \succ x_0 \overline{y}$

$$\boxed{x_1 \succ x_2 \succ x_0} \qquad \boxed{y \succ \overline{y}}$$

$P = \{V_1, V_2\}$: $Seq(r_1, r_2)(P) = x_0 y$

$P' = \{V_1, V_2, V_3\}$: $Seq(r_1, r_2)(P') = x_1 \overline{y}$

But 3 prefers $x_0 y$ to $x_1 \overline{y}$.

How should such a vote be conducted?

Some classes of solutions:

1. don't bother and vote separately on each variable.
2. ask voters to specify their preference relation by ranking all alternatives *explicitly*.
3. limit the number of possible alternatives that voters may vote for.
4. ask voters to report only a small part of their preference relation and appply a voting rule that needs this information only, such as plurality.
5. ask voters their preferred alternative(s) and complete them automatically using a predefined *distance*.
6. *sequential voting* : decide on every variable one after the other, and broadcast the outcome for every variable before eliciting the votes on the next variable.
7. **use a *compact preference representation language* in which the voters' preferences are represented in a concise way.**

7. **use a *compact preference representation language* in which the voters' preferences are represented in a concise way.**

\+ no domain restriction, provided the language is totally expressive.

– potentially expensive in elicitation and/or computation: computing the winner is generally NP-hard or coNP-hard.

Examples of such approaches:

- using GAI-nets: (Gonzalès & Perny, 08);
- using CP-nets: (Xia, Conitzer & Lang, 08);
- using weighted logical formulae: (Uckelman, 09).

Any preference relation on a combinatorial domain is compatible with some CP-net (possibly with cyclic dependencies).

Elicit the CP-net corresponding to each voter and aggregate “locally”.

Example 1 (swimming pool): 5 voters, 2 binary variables S, T

2 voters	2 voters	1 voter
$S\bar{T} \succ \bar{S}T \succ \bar{S}\bar{T} \succ ST$	$\bar{S}T \succ S\bar{T} \succ \bar{S}\bar{T} \succ ST$	$ST \succ \bar{S}T \succ S\bar{T} \succ \bar{S}\bar{T}$

S $\longleftrightarrow$ **T**

S	T
$T : \bar{S} \succ S$	$S : \bar{T} \succ T$
$\bar{T} : S \succ \bar{S}$	$\bar{S} : T \succ \bar{T}$

S $\longleftrightarrow$ **T**

S	T
$T : \bar{S} \succ S$	$S : \bar{T} \succ T$
$\bar{T} : S \succ \bar{S}$	$\bar{S} : T \succ \bar{T}$

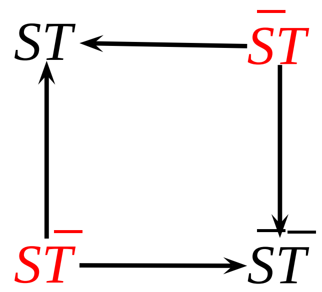
S **T**

S	T
$S \succ \bar{S}$	$T \succ \bar{T}$

apply an aggregation function (here majority) on each entry of each table

S $\longleftrightarrow$ **T**

S	T
$T : \bar{S} \succ S$	$S : \bar{T} \succ T$
$\bar{T} : S \succ \bar{S}$	$\bar{S} : T \succ \bar{T}$



Example 2: 3 voters, 2 binary variables *A*, *B*

$\mathbf{A} \longrightarrow \mathbf{B}$

$A \succ \bar{A}$

$A : B \succ \bar{B}$
$\bar{A} : \bar{B} \succ B$

$\mathbf{A} \longleftarrow \mathbf{B}$

$B : A \succ \bar{A}$
$\bar{B} : \bar{A} \succ A$

$B \succ \bar{B}$

$\mathbf{A} \qquad \mathbf{B}$

$\bar{A} \succ A$

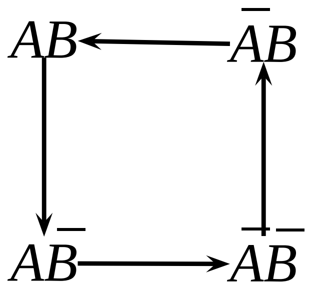
$\bar{B} \succ B$

apply an aggregation function (here majority) on each entry of each table

$\mathbf{A} \longleftrightarrow \mathbf{B}$

$B : \bar{A} \succ A$
$\bar{B} : A \succ \bar{A}$

$A : B \succ \bar{B}$
$\bar{A} : \bar{B} \succ B$



+ always applicable

– elicitation cost: in the worst case, exponential number of queries to each voter

– computation cost: dominance in CP-nets with cyclic dependencies is PSPACE-complete

– there might be no winner; there might be several winners

How should such a vote be conducted?

Some classes of solutions:

1. don't bother and vote separately on each variable.
2. ask voters to specify their preference relation by ranking all alternatives *explicitly*.
3. limit the number of possible alternatives that voters may vote for.
4. ask voters to report only a small part of their preference relation and appply a voting rule that needs this information only, such as plurality.
5. ask voters their preferred alternative(s) and complete them automatically using a predefined *distance*.
6. *sequential voting* : decide on every variable one after the other, and broadcast the outcome for every variable before eliciting the votes on the next variable.
7. use a *compact preference representation language* in which the voters' preferences are represented in a concise way.

Conclusion: *either impose a strong domain restriction, or pay a high communication and computational cost*

1. Introduction to computational social choice
2. Background
3. Topic 1: computationally hard voting rules
4. Topic 2: voting on combinatorial domains
5. **Topic 3: computational aspects of strategyproofness**
6. Topic 4: communication and incomplete knowledge
7. Topic 5: other issues

Manipulation and strategyproofness

Gibbard (73) and Satterthwaite (75) ’s theorem: if the number of candidates is at least 3, then any nondictatorial, surjective voting rule is manipulable for some profiles.

Barriers to manipulation:

- making manipulation *less efficient*: make as little as possible of the others’ votes known to the would-be manipulating coalition
- make manipulation *hard to compute*.
  First papers on the topic: Bartholdi, Tovey & Trick (89); Bartholdi & Orlin (91); then Conitzer and Sandholm (02), and lots of papers since then.

Complexity of manipulation

- 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?

Complexity of manipulation

Manipulating the Borda rule by a single voter

a	b	c	d
b	a	e	c
d	e	a	b
c	d	b	a
e	c	d	e

Current Borda scores:

a: 10

b: 10

c: 8

d: 7

e: 5

Is there a constructive manipulation by one voter for a? for b? for c? for d? for e?

Complexity of manipulation

Manipulating the Borda rule by two voters

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 manipulation

Example: 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}, ..., and finally x_2 in the bottom position.

If x_1 does not becomes a winner then there exists no manipulation for x.

Consequence: for Borda, CONSTRUCTIVE MANIPULATION EXISTENCE BY ONE VOTER is in P. (Bartholdi, Tovey & Trick, 89).

- manipulation by coalitions of more than one voter: *open problem*
- NP-complete for *weighted voters*, even for 3 candidates (Conitzer & Sandholm, 2002)

Complexity of (unweighted) manipulation

From Xia *et al.* (09):

Number of manipulators	1	at least 2
Copeland	P (1)	NP-complete (2)
STV	NP-complete (3)	NP-complete (3)
veto	P (4)	P (4)
cup	P (5)	P (5)
maximin	P (1)	NP-complete (6)
ranked pairs	NP-complete (6)	NP-complete (6)
Bucklin	P (6)	P (6)
Borda	P (1)	?

(1) Bartholdi *et al.* (89); (2) Falisezwski *et al.* (08); (3) Bartholdi and Orlin (91);
(4) Zuckerman *et al.* (08); (5) Conitzer *et al.* (07); (6) Xia *et al.* (09).

Complexity of manipulation

An important concern:

- a worst-case NP-hardness results only says that *sometimes* (maybe rarely), computing a manipulation will be hard
  $\Rightarrow$ too weak
- a few preliminary *negative* results about the average hardness of manipulation (Conitzer and Sandholm, 06; Procaccia and Rosenschein, 07).

Other kinds of strategic behaviour: **procedural control**

Some voting procedures can be controlled by the authority conducting the election (i.e. the chair) to achieve strategic results.

Several kinds of control:

- adding / deleting / partitioning candidates
- adding / deleting / partitioning voters

For each type of control and each voting rule r, three possivilities

- r is *immune to control*: it is never possible for the chairman to change a dandidate c from a non-winner to a unique winner.
- r is *resistant to control*: r is not immune and it is computationally hard to recognize opportunities for control
- r is *vulnerable to control*: r is not immune and it is computationally easy to recognize opportunities for control

Other kinds of strategic behaviour: **bribery**

GIVEN: a set C of candidates, a set $V = \{1, \ldots, n\}$ of voters specified with their preferences, n integers $p_1, \ldots, p_n$ (p_i = price for making voter i change his vote), a distinguished candidate c, and a nonnegative integer K.

QUESTION: Is it possible to make c a winner by changing the preference lists of voters while spending at most K?

(Rothe, Hemaspaandra and Hemaspaandra, 07):

- for plurality: BRIBERY is in P (and NP-complete for weighted voters)
- for approval voting: BRIBERY is in NP-complete, even for unit prices ($p_i = 1$ for each i)

variations on bribery: nonuniform bribery (Faliszewski, 08), swap bribery (Elkind, Faliszewski an Slinko, 09)

1. Introduction to computational social choice
2. Background
3. Topic 1: computationally hard voting rules
4. Topic 2: voting on combinatorial domains
5. Topic 3: computational aspects of strategyproofness
6. **Topic 4: communication and incomplete knowledge**
7. Topic 5: other issues

Incomplete knowledge and communication complexity

Given some *incomplete* description of the voters' preferences,

- is the outcome of the voting rule determined?
- if not, whose information about which candidates is needed?

4 voters: $c \succ d \succ a \succ b$

2 voters: $a \succ b \succ d \succ c$

2 voters: $b \succ a \succ c \succ d$

1 voter: $? \succ ? \succ ? \succ ?$

plurality ?

Borda ?

Incomplete knowledge and communication complexity

Given some *incomplete* description of the voters' preferences,

- is the outcome of the voting rule determined?
- if not, whose information about which candidates is needed?

4 voters: $c \succ d \succ a \succ b$

2 voters: $a \succ b \succ d \succ c$

2 voters: $b \succ a \succ c \succ d$

1 voter: $? \succ ? \succ ? \succ ?$

plurality winner already known (c)

Borda ?

Incomplete knowledge and communication complexity

Given some *incomplete* description of the voters' preferences,

- is the outcome of the voting rule determined?
- if not, whose information about which candidates is needed?

4 voters: $c \succ d \succ a \succ b$

2 voters: $a \succ b \succ d \succ c$

2 voters: $b \succ a \succ c \succ d$

1 voter: $? \succ ? \succ ? \succ ?$

plurality winner already known (c)

Borda

partial scores (for 8 voters): a: 14 ; b: 10 ; c: 14; d: 10

$\Rightarrow$ only need to know the last voters's preference between a and c

Incomplete knowledge and communication complexity

More general problems to be considered:

- Which elements of information should we ask the voters and when on order to determine the winner of the election while minimizing communication?
- When the votes are only partially known: is the winner already determined? Which candidates can still win?
- When only a part of the electorate have expressed their votes, how can we synthesize the information expressed by this subelectorate as succinctly as possible?
- When the voters have expressed their votes on a set of candidates and then some new candidates come in, who among the initial candidates can still win?
- How should sincerity and strategyproofness be reformulated when agents express incomplete preferences?

Possible and necessary winners

More generally: *incomplete knowledge* of the voters' preferences.

For each voter: 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.

Given a voting rule r, an incomplete profile P, and a candidate c:

- 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

$a \succ b, a \succ c$	$b \succ a$	$c \succ a \succ b$	plurality with tie-breaking priority $b > a > c$	Condorcet
abc	*cba*	*cab*	c	c
abc	*bca*	*cab*	b	-
abc	*bac*	*cab*	b	a
acb	*cba*	*cab*	c	c
acb	*bca*	*cab*	b	c
acb	*bac*	*cab*	c	a

Possible Condorcet winners: $\{a, c\}$; possible plurality$_{b>a>c}$-winners: $\{b, c\}$.

Possible and necessary winners

- Konczak & Lang, 05: definitions and first (partly wrong) complexity results
- Walsh, 07; Pini et al., 07: specific study
- Xia & Conitzer, 08: complexity results for most common voting rules
- Betzler, Hemmann & Niedermeyer: parameterized complexity
- Betzler & Dorn, 09: complexity results for (almost) all scoring rules
- Faliszewski et al., 09: swab bribery, generalizing the possible winner problem.

Possible and necessary winners

Two particular cases:

possible/necessary winners with respect to addition of voters

A subset of voters A have reported a full ranking; the other ones have not reported anything.

Links with coalitional manipulation:

- x is a possible winner if the coalition $N \setminus A$ has a constructive manipulation for x.
- x is a necessary winner if the coalition $N \setminus A$ has no destructive manipulation against x.

possible/necessary winners with respect to addition of candidates

The voters have reported a full ranking on a subset of candidates X (and haven't said anything about the remaining candidates).

Possible and necessary winners with respect to addition of candidates

New candidates sometimes come while the voting process is going on:

- Doodle: new dates become possible
- recruiting committee: a preliminary vote can be done before the last applicants are inrerviewed

Obviously: for any reasonable voting rule, any new candidate must be a possible winner.

Question: *who among the initial candidates can win?*

Example :

- $n = 12$ voters; initial candidates : $X = \{a, b, c\}$; one new candidate y.
- voting rule = plurality with tie-breaking priority $a > b > c > y$
- plurality scores before y is taken into account: $a \mapsto 5$, $b \mapsto 4$, $c \mapsto 3$.

Who are the possible winners?

Possible and necessary winners with respect to addition of candidates

General result for plurality: if P_X is the profile, X the initial candidates, $ntop(P_X, x)$ the number of voters who rank x in top position in P_X; then: $x \in X$ is a possible winner for P_X with respect to the addition of k new candidates *iff*

$$ntop(P_X, x) \geq \frac{1}{k} . \sum_{x_i \in X} \max(0, ntop(P_X, x_i) - ntop(P_X, x))$$

where $ntop(P_X, x)$ is the plurality score of x in P_X.

Possible and necessary winners with respect to addition of candidates

Example 2 :

- $n = 4$ voters; initial candidates : $X = \{a, b, c, d\}$; k new candidates $y_1, \ldots, y_k$.
- voting rule = Borda
- initial profile: $P = \langle bacd, bacd, bacd, dacb \rangle$.
  Borda scores: $a \mapsto 8$, $b \mapsto 9$, $c \mapsto 4$, $d \mapsto 3$.

Who are the possible winners, depending on the value of k?

Possible and necessary winners with respect to addition of candidates

Example 2 :

- $n = 4$ voters; initial candidates : $X = \{a, b, c, d\}$; k new candidates $y_1, \ldots, y_k$.
- voting rule = Borda
- initial profile: $P = \langle bacd, bacd, bacd, dacb \rangle$.

A useful lemma: x is a possible winner for P_X w.r.t. the addition of k new candidates if and only if x is the Borda winner for the profile on $X \cup \{y_1, \ldots, y_k\}$ obtained from P_X by putting $y_1, \ldots, y_k$ right below x (in an arbitrary order) in every vote of P_X.

Who are the possible winners, depending on the value of k??

- for any $k \geq 1$, a and b are possible winners;
- for any $k \geq 5$, a, b and d are possible winners;
- for any value of k, c is not a possible winner.

More general results in (Chevaleyre *et al.*, 10).

Introduction to protocols and communication complexity

Two key references:

- A.C Yao, Some complexity questions related to distributed computing, Proc. 11th ACM Symposium on Theory of Computing, 1979, 209-213
- E. Kushilevitz and N. Nisan, Communication complexity, Cambridge University Press, 1997.

Communication problem: a set of n agents has to compute a function $f(x_1,\ldots,x_n)$ given that the input is distributed among the agents: initially, agent 1 knows only x_1, $\ldots$, and agent n knows x_n.

Protocol: binary tree where each node is labelled with an agent and an action policy specifying a bit the agent should communicate, depending on her knowledge.

Informally: a protocol is similar to an algorithm, except that instructions are replaced by communication actions between agents, and such that communication actions are based on the *private information* of the agents.

Communication complexity of voting rules

From (Conitzer & Sandholm, 05).

- *Voting rule*

  $$r : \mathcal{P}^n \longrightarrow \mathcal{X}$$

  A voting rule does not specify how the votes are elicited from the voters by the central authority.

- *Protocol for a voting rule r*
  Communication protocol for computing $r(V_1, \ldots, V_n)$, given that V_i is the private information of agent (voter) i.

- *Communication complexity of a voting rule r*: minimum cost of a protocol for r.

Communication complexity of voting rules

A protocol for any voting rule r:

step 1 every voter i sends V_i to the central authority

$\hookrightarrow n \log(p!)$ bits

step 2 [the central authority sends back the name of the winner to all voters]

$\hookrightarrow n \log p$ bits

Corollary The communication complexity of an arbitrary voting rule r is at most $n. \log(p!) [+ n \log p]$

From now on, we shall ignore step 2.

Communication complexity of voting rules

Example 1: plurality

A simple protocol:

voters send the name of their most preferred candidate to the central authority

$\hookrightarrow n \log p$ bits

Corollary The communication complexity of plurality is at most $n.\log p$

Communication complexity of voting rules

Obtaining a lower bound: via the *fooling set* technique.

Details on request (off-line)

Proposition: the communication complexity of plurality with runoff is in $\Theta(n.\log p)$ (Conitzer & Sandholm, 05)

Communication complexity of voting rules

Example 2: plurality with runoff.

A protocol:

step 1 voters send the name of their most preferred candidate to the central authority

$\hookrightarrow$ $\mathbf{n} \log \mathbf{p}$ bits

step 2 the central authority sends the names of the two finalists to the voters

$\hookrightarrow$ $\mathbf{2n} \log \mathbf{p}$ bits

step 3 voters send the name of their preferred finalist to the central authority

$\hookrightarrow$ $\mathbf{n}$ bits

total $\mathbf{n}(\mathbf{3} \log \mathbf{p} + \mathbf{1})$ bits (in the worst case)

Corollary: the communication complexity of plurality with runoff is in $O(n. \log p)$.

The lower bound matches:

Proposition: the communication complexity of plurality with runoff is in $\Theta(n. \log p)$ (Conitzer & Sandholm, 05)

Communication complexity of voting rules

Example 3: Single Transferable Vote (STV): a protocol

step 1 voters send their most preferred candidate to the central authority (C)

$\hookrightarrow$ $\mathbf{n} \log \mathbf{p}$ 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

$\hookrightarrow$ $\mathbf{2\frac{n}{p}} \log \mathbf{p}$ bits

step 3 similarly with the new candidate y to be eliminated. At most $\frac{n}{p-1}$ voters voted for y

$\hookrightarrow$ $\mathbf{2\frac{n}{p-1}} \log \mathbf{p}$ bits

etc.

total $\leq 2n \log p(1 + \frac{1}{p} + \frac{1}{p-1} + \ldots + \frac{1}{2}) = O(n.(\log p)^2)$.

Lower bound matches (Conitzer & Sandholm, 05)

Incomplete knowledge and communication complexity

Example 4: **Bucklin rule**:

Let q the smallest integer such that there exists a candidate x such that more than half of the voters rank x among their q preferred candidates. (Necessarily, $1 \leq q \leq \frac{p}{2}$.)

Then the winner is the candidate ranked in the q preferred candidates by the largest number of voters.

Optimal protocol for Bucklin?

Compilation complexity

From the following paper: *Y. Chevaleyre, J. Lang, N. Maudet and G. Ravilly-Abadie, Proc. IJCAI-09.*

Context: sometimes the votes do not come all together at the same time

- votes of the citizens living abroad known only a few days after the rest of the votes;
- choosing a date for a meeting: some participants vote later than others.

$\Rightarrow$ preprocess the information given by the subelectorate so as to prepare the ground for the time when the last votes are known, using *as little space as possible*.

Input only $m \leq n$ votes have been expressed.
$P = \langle V_1, \ldots, V_m \rangle$ = corresponding *partial* prole.

Question what is the minimal size needed to compile P, while still being able to compute r when the last votes come in?

A context where it is useful to compile the vote of a subelectorate: *verification of the outcome of a vote by the population*.

- the electorate is split into different districts; each district counts its ballots separately and communicates the outcome to the Ministry of Innner Affairs, which, after gathering the outcomes from all districts, determines the final outcome;
- in each district, the voters can check that the local results are sound;
- local results are made public and voters can check the final outcome from these local outcomes.

space needed to synthesize the votes of a district

= amount of information the district has to send to the central authority

If this amount of information is too large, it is impractical to publish the results locally, and therefore, difficult to check the final outcome and voters may be reluctant to accept the voting rule.

Compilation complexity

$\rho(\sigma(P), R) = r(P \cup R)$

σ compilation function

Example: r_B = Borda.

$\sigma(P)$: vector of partial Borda scores $\langle s_B(x \mid P)\rangle_{x \in X}$

$P = \langle abc, abc, cba, bca\rangle \mapsto \sigma(P) = \langle a : 3; b : 5; c : 3\rangle$.

$\rho(\sigma(P), R) = \text{argmax}_{x \in X}\,(s_B(x \mid P) + s_B(x \mid R))$

$R = \langle cab, abc\rangle \mapsto \langle a : 3+3; b : 5+1; c : 3+2\rangle \mapsto \rho(\sigma(P), R) = b$.

Compilation complexity

Size of a compilation function

Let σ be a compilation function for r

$$Size(\sigma) = \max\{|\sigma(P)| \mid P \text{ partial profile}\}$$

Compilation complexity of r:

$$C(r) = \min\{Size(\sigma) \mid \sigma \text{ compilation function for } r\}$$

$C(r)$ is the minimum space needed to compile the partial profile P

Compilation complexity and one-round communication complexity

One-round communication complexity :

- two agents A and B have to compute a function f.
- each of them knows only a part of the input.
- *one-round protocol*: A sends only one message to B, and then B sends the output to A.
- *one-round communication complexity* of f: worst-case number of bits of the best one-round protocol for f.

One-round communication complexity $\approx$ compilation complexity

- A = set of voters having already expressed their votes
- B = set of remaining voters;
- compilation of the votes of A = information that A must send to B.
- minor difference: B does not send back the output to A.

Equivalent profiles for a voting rule

r voting rule;
k number of remaining voters.

Two partial profiles P and Q are *equivalent for r* if no matter the remaining votes, they will lead to the same outcome:

$$\text{for every } R \text{ we have } r(P \cup R) = r(Q \cup R)$$

Example: r_P = plurality with tie-breaking priority $b > a > c$.

- $\langle abc, abc, bac, bac \rangle$ and $\langle acb, acb, bca, bca \rangle$ are equivalent for r_P;
- $P_1 = \langle abc, abc \rangle$ and $P_2 = \langle abc, bac \rangle$ are not equivalent for r_P: take $R = \langle bca, bca \rangle$, then $r_P(P_1 \cup R) = a \neq r_P(P_2 \cup R) = b$.

A useful result (similar result in (Kushilevitz & Nisan, 97)):

- r voting rule.
- m number of initial voters
- p number of candidates.

If the equivalence relation for r has $g(m,p)$ equivalence classes then

$$C(r) = \lceil \log g(m,p) \rceil$$

Corollary:

- for any voting rule r, $C(r) \leq m \log(p!)$;
- for any anonymous voting rule r, $C(r) \leq \min(m \log(p!), p! \log m)$.
- the compilation complexity of a dictatorship is $\log p$;
- the compilation complexity of r is 0 if and only if r is constant.

Compilation complexity of voting rules:

- *plurality*: P and P' are equivalent *iff* for all x, $ntop(P,x) = ntop(P',x)$, where $ntop(P,x)$ be the number of votes in P ranking x first.
- *Borda*: P and P' are equivalent *iff* for all x, $score_B(x,P) = score_B(x,P')$, where $score_B(x,P)$ = Borda score of x obtained from the partial profile P
- *rules based on the majority graph*: For any Condorcet-consistent rule based on the (unweighted/weighted) majority graph, P and P' are equivalent *iff* $\mathcal{M}_P = \mathcal{M}_{P'}$, where $\mathcal{M}_P$ is the *weighted* majority graph associated with P.
- *plurality with runoff*: P and Q are equivalent *iff* these two conditions hold: (a) for every x, $ntop(P,x) = ntop(Q,x)$; and (b) $\mathcal{M}_P = \mathcal{M}_Q$.
- *STV*: P and Q are equivalent *iff* for all subset of candidates V and $x \in V$, $ntop(P_{-V},x) = ntop(Q_{-V},x)$
  (For any set of candidates that can possibly be eliminated, the plurality scores of the remaining candidates must be the same in P and Q.)

Results on compilation complexity follow from these results by computing bounds on the number of equivalence classes.

Incomplete knowledge and communication complexity

Other issues:

- voting with partial ballots: strategical issues (Pini *et al.*, 07; Endriss *et al.*, 09)
- communication issues with single-peaked preferences (Trick, 89; Doignon, 05; Conitzer, 08; Escoffier *et al.*, 08)
- sequential announcements of votes (Pacuit and Parikh, 06; Airiau and Endriss, 09; Elkind *et al.*, 09; Xia and Conitzer, 10)

What if there were one more lecture?

Learning voting rules (Procaccia et al., 07/08)

Given a family F of voting rules (for instance: scoring rules) and a set of examples (P, x) where P is a profile and x the winning candidate, find a voting rule in F fitting the examples as much as possible.

Robustness of voting rules (Procaccia, Rosenschein & Kaminka, 07)

- r voting rule, $k \in \mathbb{N}$, P.
- elementary change = permutation of two adjacent candidates in a voter's preferences;
- $D_k(P)$ = set of profiles obtained from P by k elementary changes.
- k-robustness of r for P: $\rho_k(r, P)$ = probability that $r(P') = r(P)$ where P' is chosen according to a uniform law on $D_k(P)$.
- k-robustness of r: $\rho_k(r) = \min_P \rho_k(r, P)$

Group plannning *e.g.*, Ephrati & Rosenschein (93)

etc.