Computational Social Choice: Autumn 2013

Ulle Endriss
Institute for Logic, Language and Computation
University of Amsterdam

Plan for Today

Elections often have a *combinatorial structure*:

- Electing a committee of k members from amongst n candidates.
- Voting on n propositions (yes/no) in a referendum.

Clearly, the number of alternatives can quickly become *very large*.
So we face both a *choice-theoretic* and a *computational challenge*.

Today we will highlight some of the problems associated with voting in combinatorial domains and introduce several of the approaches that have been proposed to address them.

More details are in the expository paper by Chevaleyre et al. (2008).
See also Section 4 in *Logic and Social Choice Theory*.

Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet. Preference Handling in Combinatorial Domains: From AI to Social Choice. *AI Magazine*, 29(4):37–46, 2008.

U. Endriss. Logic and Social Choice Theory. In A. Gupta and J. van Benthem (eds.), *Logic and Philosophy Today*, College Publications, 2011.

The Paradox of Multiple Elections

13 voters are asked to each vote *yes* or *no* on three issues:

- 3 voters each vote for YNN, NYN, NNY.
- 1 voter each votes for YYY, YYN, YNY, NYY.
- No voter votes for NNN.

If we use the *simple majority* rule *issue-by-issue*, then NNN wins, because on each issue 7 out of 13 vote *no*.

This is an instance of the *paradox of multiple elections:* the winning combination received not a single vote!

S.J. Brams, D.M. Kilgour, and W.S. Zwicker. The Paradox of Multiple Elections. *Social Choice and Welfare*, 15(2):211–236, 1998.

What's a Paradox?

Before we start: Why did we call this a *paradox*?

We can give a *general definition of paradox*, consisting of:

- an aggregation rule F,
- a profile of ballots $\boldsymbol{B}$, and
- an integrity constraint IC (a property applicable to both ballots and outcomes, such as $\neg(\neg X \wedge \neg Y \wedge \neg Z)$).

Such a triple $(F, \boldsymbol{B}, IC)$ is a paradox *iff* each ballot in $\boldsymbol{B}$ satisfies IC, but the outcome $F(\boldsymbol{B})$ does not.

(Observe that this definition also covers, say, the Condorcet Paradox.)

U. Grandi and U. Endriss. Binary Aggregation with Integrity Constraints. Proc. IJCAI-2011.

Voting in Combinatorial Domains

The problem of *voting in combinatorial domains:*

- Domain: variables $X_1, \ldots, X_p$ with finite domains $D_1, \ldots, D_p$
- Voters have preferences over set of combinations $D_1 \times \cdots \times D_p$.
- What should be the winning combination in $D_1 \times \cdots \times D_p$?

Today we focus on *binary* variables: $D_k = \{x_k, \bar{x}_k\}$.

Question: We have seen that voting issue-by-issue can lead to paradoxical outcomes. What other approaches are there?

Approach 1: Plurality on Combinations

Idea: Vote for combinations directly: ask each voter for her most preferred combination and apply the plurality rule.

This avoids the paradox we have seen and is computationally light.

Problem: This may lead to almost "random" decisions, unless domains are fairly small and there are many voters.

Example: Suppose there are 10 binary issues and 20 voters. Then there are $2^{10} = 1024$ combinations to vote for. Under the plurality rule, chances are very high ($\sim 83\%$) that no combination receives more than one vote (so the tie-breaking rule decides everything).

Remark: Similar comments apply for other voting rules that only elicit a small part of the voter preferences (e.g., k-approval with small k).

Approach 2: Other Rules on Combinations

Idea: Vote for combinations directly, using your favourite voting rule with the full set of combinations as the set of alternatives.

If we use a voting rule that elicits more information than the plurality rule, then we can avoid the arbitrariness problem noted before.

Problem: This will only be possible for very small domains, certainly when the voting rule requires a complete ranking of all the candidates (such as the Borda rule).

Example: Suppose there are six binary issues. This makes $2^6 = 64$ possible combinations. Hence, under the Borda rule, each voter has to choose from amongst $64! \approx 1.27 \cdot 10^{89}$ possible ballots.

Approach 3: Preselect Admissible Combinations

Idea: Select a small number of combinations and then use your favourite voting rule to elect a winner from amongst those.

Problem: Who selects the admissible combinations available for election? It is not at all clear what criteria should we should use here. This gives the chooser (probably the election chair) undue powers and opens up new opportunities for controlling elections.

Approach 4: Distance-based Aggregation

Idea: Elicit preferred choices issue-by-issue (as in the paradox), but find a better way to aggregate this information.

Distance-based approaches are promising candidates:

- Define a *distance* metric on ballots (0-1 vectors).
- Extend it to measure distance of a ballot/outcome to a profile.
- Choose the outcome that *minimises* the distance to the profile.
- Possibly restrict attention to outcomes from some *admissible set*.

Next we will see several examples.

Example: The Minimax Rule

Brams et al. (2007) propose to elect the combination that minimises the *maximal Hamming distance* to any of the voter ballots:

- Distance between two vectors = no. of issues on which they differ
- Distance between vector and profile = maximum of distances
- Admissible set of outcomes = all outcomes

That is, if your unhappiness is proportional to the number of issues on which you do not get your way, then the *minimax rule* maximises the happiness of the unhappiest voter.

Compare this to the "*minisum rule*": choose the outcome that minimises the *sum* of the *Hamming distances* to the individual ballots (insight: this is just issue-by-issue majority!).

S.J. Brams, D.M. Kilgour, and M.R. Sanver. A Minimax Procedure for Electing Committees. *Public Choice*, 132:401–420, 2007.

Consistent Distance-based Aggregation

We may also decide to choose from some *admissible set* of possible outcomes the combination that minimises the distance to the profile:

- The admissible set could be all outcomes that meet a certain *integrity constraint* (e.g., "say YES at least once").
- If we don't know what possible integrity constraints our voters might want to see respected, the best we can do is to equate the admissible set with the set of ballots received $\rightsquigarrow$ generalised dictatorships/*representative-voter rules* (Endriss & Grandi, 2013).

U. Endriss and U. Grandi. Binary Aggregation by Selection of the Most Representative Voter. Proc. MPREF-2013.

Approach 5: Sequential Voting

Idea: Vote separately on each issue, but do so sequentially to give voters the opportunity to make their vote for one issue dependent on other issues already decided upon.

We will discuss two basic results for this approach:

- A simple result showing that sequential voting does address some of the problems raised by the multiple election paradox.
- A stronger result in case we can make the assumption that voter preferences are induced by "compatible" CP-nets.

Condorcet Losers

A *Condorcet loser* is a candidate that loses against any other candidate in a pairwise contest. Electing a CL is very bad.

But the plurality rule will sometimes elect the CL:

2 voters report:	$x \succ y \succ z$
2 voters report:	$y \succ x \succ z$
3 voters report:	$z \succ x \succ y$

Here, z is both the plurality winner and the Condorcet loser.

Remark: Another interpretation of our original multiple election paradox is that NNN could have been the Condorcet loser.

Sequential Voting and Condorcet Losers

Lacy and Niou (2000) show that sequential voting avoids the problem of electing Condorcet losers:

Proposition 1 *Sequential voting (with plurality) over binary issues never results in a winning combination that is a Condorcet loser.*

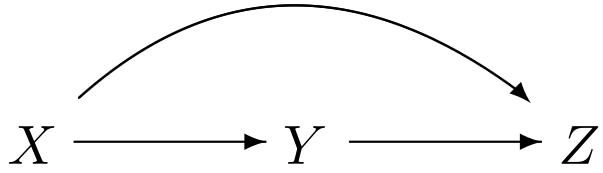
Proof: Just think what happens during the election for the final issue. The winning combination cannot be a Condorcet loser, because it does, at least, win against the other combination that was still possible after the penultimate election. ✓

A stronger requirement is *Condorcet consistency:* elect the *Condorcet winner* whenever it exists. Sequential voting *cannot* guarantee this.

D. Lacy and E.M.S. Niou. A Problem with Referendums. *Journal of Theoretical Politics*, 12(1):5–31, 2000.

Preferential Dependency Graphs

A preference order induces a *preferential dependency graph* on issues: issue Y depends on X if there exist situations where you need to know the value of X before you can decide on your preference regarding Y.



Remark: *CP-nets*, an important language for preference representation in Artificial Intelligence with several applications in COMSOC, are directly based on this idea: dependency graph + table indicating local preferences over each variable given choices made for its parents.

C. Boutilier, R.I. Brafman, C. Domshlak, H.H. Hoos, and D. Poole. CP-nets: A Tool for Representing and Reasoning with Conditional *Ceteris Paribus* Preference Statements. *Journal of Artificial Intelligence Research*, 21:135–191, 2004.

Sequential Voting and Condorcet Winners

Suppose that there exists a linear order on the issues that is compatible with each voter's preferential dependency graph (e.g., they might all have different CP-nets, but share the same underlying *acyclic* graph).

Idea: Vote sequentially in this order!

Lang and Xia (2009) have shown (proof omitted but easy):

Proposition 2 *Under above assumptions, sequential voting is Condorcet-consistent whenever all of the local voting rules are.*

This is useful, particularly when the issues are binary (as then any reasonable local rule will be Condorcet-consistent).

J. Lang and L. Xia. Sequential Composition of Voting Rules in Multi-issue Domains. *Mathematical Social Sciences*, 57(3):304–324, 2009.

Approach 6: Combinatorial Vote

Idea: Ask voters to report their ballots using a compact preference *representation language* and apply your favourite voting rule to the succinctly encoded ballots received.

Lang (2004) calls this approach *combinatorial vote*.

Discussion: A promising approach, but not too much is known to date about what would be good choices for preference representation languages or voting rules, or what algorithms to use to compute the winners. Also, complexity can be expected to be very high.

J. Lang. Logical Preference Representation and Combinatorial Vote. *Annals of Mathematics and Artificial Intelligence*, 42(1–3):37–71, 2004.

The Language of Prioritised Goals

Consider a *binary* comb. domain $\mathcal{D} = D_1 \times \cdots \times D_p$, with $|D_i| = 2$. Associate $\mathcal{D}$ with a set $PS = \{X_1, \ldots, X_p\}$ of propositional variables, i.e., identify truth assignments for PS (models) with elements of $\mathcal{D}$.

Use propositional formulas to express *goals* and use *numbers* to indicate their importance. This induces a weak order:

> Suppose (φ, k_1) has higher *priority* than (ψ, k_2) if $k_1 > k_2$.
>
> Under the *lexicographic* form of aggregation, we prefer M to M' if there exists a k such that for all $j > k$ both M and M' satisfy the same number of goals of rank j, and M satisfies more goals of rank k.

Other forms of aggregation are possible.

J. Lang. Logical Preference Representation and Combinatorial Vote. *Annals of Mathematics and Artificial Intelligence*, 42(1–3):37–71, 2004.

Combinatorial Vote: Example

Use the language of *prioritised goals* (1 has higher priority than 0) with *lexicographic aggregation* together with the *Borda rule:*

- Voter 1: $\{X{:}1,\, Y{:}0\}$ induces order $xy \succ_1 x\bar{y} \succ_1 \bar{x}y \succ_1 \bar{x}\bar{y}$
- Voter 2: $\{X \vee \neg Y{:}0\}$ induces order $x\bar{y} \sim_2 xy \sim_2 \bar{x}\bar{y} \succ_2 \bar{x}y$
- Voter 3: $\{\neg X{:}0,\, Y{:}0\}$ induces order $\bar{x}y \succ_3 \bar{x}\bar{y} \sim_3 xy \succ_3 x\bar{y}$

As the induced orders need not be strict linear orders, we use a *generalisation of the Borda rule:* an alternative gets as many points as she dominates other alternatives. So we get these Borda scores:

$$xy : 3+1+1=5 \qquad \bar{x}y : 1+0+3=4$$
$$x\bar{y} : 2+1+0=3 \qquad \bar{x}\bar{y} : 0+1+1=2$$

So combinatorial alternative xy wins.

Combinatorial vote *proper* would be to compute the winner *directly* from the goalbases, without the detour via the induced orders.

Single Goals and Generalised Plurality

Next a complexity result exemplifying the limitations of the approach.

We will work with the following language and voting rule:

- Using the *language of single goals*, each voter specifies just one goal (an arbitrary propositional formula) with priority 1.
- Under the *generalised plurality rule*, a voter gives 1 point to each undominated alternative.

Here are two examples, for the set of variables $\{X, Y\}$:

- The goal $\neg X \wedge Y$ induces the order $\bar{x}y \succ xy \sim x\bar{y} \sim \bar{x}\bar{y}$, so only combination $\bar{x}y$ receives 1 point.
- The goal $X \vee Y$ induces the order $xy \sim \bar{x}y \sim x\bar{y} \succ \bar{x}\bar{y}$, so combinations xy, $\bar{x}y$, $x\bar{y}$ receive 1 point each.

Winner Verification under Plurality

Define the following decision problem, for a preference representation language $\mathcal{L}$ and a voting rule F:

$\textsc{Among-Winners}(\mathcal{L}, F)$
Instance: Profile $\boldsymbol{R}$ expressed in $\mathcal{L}$; combination $x^\star$.
Question: Is $x^\star \in F(\boldsymbol{R})$?

The following result is due to Lang (2004):

Proposition 3 $\textsc{Among-Winners}$ *is coNP-complete for the language of single goals and the generalised plurality rule.*

Recall that coNP is the complement of the complexity class NP (i.e., it is the complexity class of checking validity in propositional logic).

J. Lang. Logical Preference Representation and Combinatorial Vote. *Annals of Mathematics and Artificial Intelligence*, 42(1–3):37–71, 2004.

Proof

We show, equivalently to the claim, that checking whether $x^\star$ is *not* a winner under plurality is NP-complete:

- NP-membership: Let $\varphi_1, \ldots, \varphi_n$ be the goals of the voters. The plurality score of any candidate x can be computed by adding one point for each voter i with $x \models \varphi_i$ or φ_i being inconsistent (the two cases in which x is undominated). But to *compare* two candidates x and y, we only need to check $x \models \varphi_i$ and $y \models \varphi_i$, which we can do in polynomial time. Hence, if someone claims that $x^\star$ is *not* a winner and names a stronger candidate x, then this can be verified in polynomial time. ✓
- NP-hardness: By reduction from SAT. Let φ a formula for which we want to check satisfiability. Let p be a new propositional symbol; then φ is satisfiable iff $\varphi \wedge p$ is. Create a single voter with goal $\varphi \wedge p$. Pick candidate combination $x^\star$ with $x^\star \not\models p$ (must exist), i.e., also $x^\star \not\models \varphi \wedge p$. Then $x^\star$ is *not* a plurality winner iff there exists another candidate x with $x \models \varphi \wedge p$, i.e., iff $\varphi \wedge p$ is satisfiable. ✓

Summary

We have seen several approaches for tackling the problem of voting in *combinatorial domains* (i.e., voting in multi-issue elections).

To date, no clear solution has emerged. Good candidates:

- Distance-based approaches
- Sequential voting
- Voting with compactly expressed preferences

Any approach has to balance a *choice-theoretic challenge* (eliciting too little information from voters leads to paradoxes) and a *computational challenge* (eliciting too much information may be intractable).

What next?

Our next topic will be *judgment aggregation*, which deals with the aggregation of judgments regarding the truth of propositional formulas.

- There are close connections to today's topic, social choice in *combinatorial domains* (the set of all truth assignments being the combinatorial domain in question).
- A second connection to the first part of the course is that preference aggregation can be embedded into JA.

Later on in the course we will discuss *fair division*, which also gives rise to combinatorial optimisation problems (but now for a different notion of preference, usually modelled in terms of utility functions).