Plan for Today

Besides the complexity-theoretic properties of voting procedures, another computational concern in voting is raised by the fact that the alternatives to vote for often have a combinatorial structure:

- Electing a committee of $k$ members from amongst $n$ candidates.
- During a referendum (in Switzerland, California, places like that), voters may be asked to vote on several propositions.

Clearly, the number of alternatives can quickly become very large. Can we somehow exploit the internal structure of the alternatives?

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

The Paradox of Multiple Elections

Suppose 13 voters are asked to each vote yes or no on three issues; and we use the plurality rule for each issue independently to select a winning combination:

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

But then NNN wins: 7 out of 13 vote no on each issue.

This is an instance of the paradox of multiple elections: the winning combination receives the fewest number of votes.

Above is the smallest instance of the paradox (in its strict form). Clearly, this kind of phenomenon will be very frequent.

Possible Solutions

(1) Use your favourite voting rule with the full set of combinations as the set of candidates.

(2) Do as for (1) but only require voters to rank their $k$ most preferred combinations (e.g., for plurality we’d have $k = 1$).

(3) Select a small number of combinations and then use your favourite voting rule to elect a candidate from amongst those.

(4) Ask voters to report their preferences using a compact representation language and apply your favourite voting rule to the succinctly encoded ballots received (“combinatorial vote”).

(5) Vote separately on each issue (as in the paradox), but —
   (a) identify conditions under which the paradox can be avoided;
   (b) find a novel way of aggregating the votes to select a winner;
   (c) do so sequentially rather than simultaneously.

Solution (1): Explicit Vote for Combinations

Idea: Vote for combinations directly: use your favourite voting rule with the full set of combinations as the set of candidates.

Problem: This will only be possible in very small domains, in particular when the voting rule requires a complete ranking of all 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 between $64! \approx 1.27 \cdot 10^{69}$ possible ballots.

But feasibility considerations aside, this is the only approach that will be fully satisfactory from an “idealistic” point of view.

Solution (2): Vote for Top Combinations only

Idea: Vote for combinations directly, but only require voters to rank their $k$ most preferred combinations, for some small number $k$.

Specifically, for the plurality rule $k = 1$ will suffice.

This clearly addresses the communication problems of Solution (1).

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 to be decided upon and 100 voters. Then there are $2^{10} = 1024$ combinations to vote for. Under the plurality rule, chances are that no combination receives more than one vote (so the tie-breaking rule decides).

Solution (3): Vote for Selected Combinations only

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

Problem: Who selects the candidate combinations? It is not at all clear what criteria should be used here. This gives the chooser (probably the election chair) undue powers and opens up new opportunities for controlling elections.
Solution (4): Combinatorial Vote

Idea: Ask voters to report their preferences 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: This seems the most promising approach so far, although not too much is known to date 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.

Example

Use the language defined by the lexicmin ordering over prioritised goals with the Borda rule (goals are labelled by their rank):

- Voter 1: \{A\:0, B\:1\} induces order \( AB \succ 1 \ AB \succ 1 \bar{A} \bar{B} \)
- Voter 2: \{A \lor 
\neg B\:0\} induces order \( \bar{A} \bar{B} \sim 2 \ AB \sim 2 \bar{A} \bar{B} \)
- Voter 3: \{\neg A\:0, B\:0\} induces order \( \bar{A} \bar{B} \succ 3 \bar{A} \bar{B} \succ 3 \bar{A} \bar{B} \)

As the induced orders need not be strict linear orders, we use a generalisation of the Borda rule: a candidate gets as many points as she dominates other candidates. So we get these Borda counts:

\[
\begin{align*}
AB & : 3 + 1 + 1 = 5 \\
\bar{A}B & : 1 + 0 + 3 = 4 \\
\bar{A} \bar{B} & : 2 + 1 + 0 = 3 \\
\bar{A} \bar{B} & : 0 + 1 + 1 = 2
\end{align*}
\]

So combination \( AB \) wins.

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

Winner Verification under Plurality

Define the following decision problem, for some preference representation language \( \mathcal{L} \) and some voting rule \( f \):

Among-Winners(\( \mathcal{L}, f \))

Instance: Voter preferences in \( \mathcal{L} \); candidate combination \( c \).

Question: Is \( c \) amongst the winners if voting rule \( f \) is used?

The following result is due to Lang (2004):

Theorem 1 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 and the complexity class of validity checking in propositional logic.

Proof

We show, equivalently to the claim, that checking whether \( c \) is not a winner under plurality is NP-complete:

- **NP-membership:** Let \( G_1, \ldots, G_n \) 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 \( x \models G_i \) or \( G_i \) being inconsistent (the two cases in which \( x \) is undominated). To compare two candidates we only need to check \( x \models G_i \), which can be done in polynomial time. Hence, if someone claims that \( c \) 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 \( \varphi \) a formula for which we want to check satisfiability. Let \( p \) be a new propositional symbol; so \( \varphi \) is satisfiable iff \( \varphi \land p \) is. Create a single voter with goal \( \varphi \land p \). Pick state/candidate \( c \) with \( c \not\models p \) (must exist), i.e., also \( c \not\models \varphi \land p \). Then \( c \) is not a plurality winner iff there exists another candidate \( c' \) with \( c' \models \varphi \land p \), i.e., iff \( \varphi \land p \) is satisfiable. ✓

Verification of Condorcet Winners

Recall that a **Condorcet winner** is a candidate that beats every other candidate in pairwise majority contests.

**Theorem 2** **Condorcet-Winner** is coNP-complete for the language of single goals.

Other results from the same paper show that the complexity of checking the winning status of candidates for more interesting preference languages tends to be above NP/coNP.

Possible Solutions (reminder)

1. Use your favourite voting rule with the full set of combinations as the set of candidates.
2. Do as for (1) but only require voters to rank their \( k \) most preferred combinations (e.g., for plurality we’d have \( k = 1 \)).
3. Select a small number of combinations and then use your favourite voting rule to elect a candidate from amongst those.
4. Ask voters to report their preferences using a compact representation language and apply your favourite voting rule to the succinctly encoded ballots received (“combinatorial vote”).
5. Vote separately on each issue (as in the paradox), but —
   (a) identify conditions under which the paradox can be avoided;
   (b) find a novel way of aggregating the votes to select a winner;
   (c) do so sequentially rather than simultaneously.

Solution (5a): Vote Separately Anyway

**Idea:** Try to identify conditions under which voting separately on each issue does not lead to paradoxes.

**Discussion:** To be precise, on the face of it the “paradox” can never be avoided, but for some types of voter preferences this is not a worry. Recall that NNN won, even though it was nobody’s favourite combination. If NNN is everyone’s least favourite combination (which is possible in general), then this is really dramatic. But if voters only care about the individual decisions, then NNN is a reasonable outcome, as 9 out of 13 voters voted 2×N, for instance.
Separable Preferences

Voting on each issue separately seems fine (certainly for binary issues) if the voters all have separable preferences:

- Informal definition: your preferences for variable $X$ are independent from the chosen instantiation of the other variables
- Formal definition: for any variable $X$, any two values $x, x'$ of $X$, and any two instantiations $\vec{y}, \vec{y}'$ for all other variables, $(x, \vec{y}) \prec (x', \vec{y})$ entails $(x, \vec{y}') \prec (x', \vec{y}')$

Separability of preferences is a very strong assumption, which will often not be justified...

Example

Benoit and Kornhauser (1999) give a nice example showing that we cannot assume separability of preferences when voting for a committee (assembly) which will make decisions on our behalf:

“Consider an election for a three-person assembly that will reach majority rule decisions on three separate issues, each of which can be decided 0 or 1. Consider six candidates, two each at positions $x = (1, 1, 0), y = (1, 0, 1)$, and $z = (0, 1, 1)$ and a voter whose favorite outcome is $(1, 1, 1)$. The voter prefers a candidate at $z$ to one at $x$ to complete an assembly whose other members are at $x$ and $y$, but prefers a candidate at $x$ to one at $z$ to complete an assembly whose other members are at $y$ and $z.”

Solution (5b): Vote Separately and Aggregate

Idea: Vote separately on each issue. However, don’t just promote the issue-wise winners to the winning combination, but rather investigate other ways of aggregating the ballot information.

Discussion: Potentially interesting approach, if we can come up with a reasonable aggregation rule. We should choose a combination that somehow minimises the distance of the winning combination to the combinations the voters voted for (ballots).

This raises two questions:

1. What metric should we use to measure the distance between two combinations (e.g., winning combination and a ballot)?
2. How should we aggregate the distances to individual ballots?

Minisum and Minimax Procedures

A possible answer to Question (1), for binary issues:

- A natural choice for defining the distance between two combinations is the Hamming distance (= number of issues where they differ).

Two possible possible answers to Question (2):

- **Minisum Procedure**: select as winner a combination that minimises the sum of distances to the combinations submitted as ballots.
- **Minimax Procedure**: select a combination that minimises the maximal distance to any ballot. Proposed by Brams et al. (2007).

Observe that using the minisum procedure with the Hamming distance is just another way of circumscribing the standard approach: for each issue choose the value receiving the most votes.

Solution (5c): Sequential Vote

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.

We will discuss this approach in some depth:

- Two general results showing that sequential voting does address some of the problems raised by the multiple election paradox.
- Some stronger results in case we can make certain assumptions on voter preferences: separable preferences and preferences induced by CP-nets.

Sequential Voting and Condorcet Losers

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

Example: NNN (the winning combination) in the original multiple election paradox is such a Condorcet loser.

Lacy and Niou (2000) show that sequential voting can avoid this:

Theorem 3 Sequential voting (with plurality) over binary issues never results in a winning combination that is a Condorcet loser.

Proof sketch: 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.

Sophisticated Sequential Voting

For sophisticated voters (thinking ahead and acting strategically), Lacy and Niou (2000) prove an even stronger result:

Theorem 4 Sophisticated sequential voting (with plurality) over binary issues produces a Condorcet winner, whenever one exists.

Proof sketch: During the final election, clearly, if one of the two options is a Condorcet winner, then that option will win. Then reason backwards: in the penultimate election, all voters would prefer going down the path allowing them to vote for the Condorcet winner in the next round (if one exists).

Sequential Voting with Separable Preferences

The previous result only holds under very specific conditions.

- If we drop the assumption of sophisticated voting, what can we retain from the previous result?
- We also might want to drop the restriction to binary issues.
- We might not want to use plurality as the only local voting rule.

Another type of assumption to make concerns the preferences of voters. The most severe assumption here is separability.

Fact 1 If all voters have separable preferences and each local rule satisfies the Condorcet principle, then so does the global voting rule.

To what extent can we relax the assumption of separability?
CP-Nets

A CP-net over a set of variables $V = \{X_1, \ldots, X_p\}$ is a directed graph $G$ over $V$. Each node $X_i$ is annotated with a conditional preference table for that variable.

Each such table (for $X_i$) associates a total order over values of $X_i$ with each instantiation of the parents of $X_i$ in the graph.

Such orders over values of $X_i$ are extended to preference statements over combinations in a ceteris paribus fashion (for combinations that only differ in the value they assign to $X_i$).

The overall order induced by a CP-net is the transitive closure of these individual preference statements.

Here we only consider CP-nets with acyclic graphs.

Example: Dinner

Two variables: soup (fish or vegetable) and wine (red or white)

Above: graph, preference tables, induced preference order

(Note that the least preferred combination is shown on top.)

Other Properties

Lang and Xia (2009) also investigate for several other properties of voting rules (besides Condorcet-consistency) whether they transfer from the local voting rules to the global rule under sequential voting with CP-nets. Three examples:

- **Anonymity** does transfer.
- **Neutrality** does not transfer.
- **Consistency**, defined as follows, also does transfer:
  - A voting rule $f$ is consistent iff $f(P_1) = f(P_2)$ entails $f(P_1 \cup P_2) = f(P_1) = f(P_2)$.

What next?

Next week we will talk about another area of social choice theory, called **judgement aggregation**:

- Instead of voters with preferences over candidates, there are judges with beliefs on the truth of a number of propositions.
- Not every combination of judgements makes sense: e.g., if I believe in $A$ and $A \rightarrow B$, I should also believe in $B$.
- We have to aggregate the individual judgements into a collective judgement (that still makes sense).

There are a number of links to voting in combinatorial domains (links that have not been explored very much to date).