Combinatorial Voting and Planning

Héctor Palacios

Departmento de Informática
Universidad Carlos III de Madrid

Jan 2012

Voting

Voting: obtaining a socially preferred option, given preferences of the members of a group.

- Mary: *beach* $\succ$ *mountain* $\succ$ *abroad*
- John: *mountain* $\succ$ *beach* $\succ$ *abroad*
- Anna: *abroad* $\succ$ *mountain* $\succ$ *beach*

Which option should be the winner?

Social Choice Theory

- Social Choice Theory study Collective Decision Making

  How to aggregate the preferences of the members of a group to obtain a social preferences?
- Studied in Economy and Political Sciences.
- Problems: voting, preference aggregation, fair division, ...
- History:
  - Precursors in 18th Century: Condorcet, Borda.
  - Research Area since 1950s.
- How to study it? How many different procedures?

  Axiomatic approach

Social Choice Theory (II)

- Axiom/Properties:

  Anonymity (A): the result should not depend on who votes what.

- Characterisation:

  A voting rule that satisfying anonymity, . . . , is equivalent to Majority

- Impossibility/Possibility results:
  - Example: Arrow's theorem:

    There exists no preference aggregation function defined on the set of all preference profiles, satisfying Unanimity, IIA and Non-Dictatorship.

  - By the way, Kenneth J. Arrow received the Nobel Prize in Economics in 1972.

- Manipulation:

  Is it possible to get a result closer to my real preference by lying?

Computational Social Choice

Computational issues on ...

- Computing a winner
- Manipulate/Vote strategically
- . . .

Can it be done effectively?

Agenda

Voting: Formal definition

Notation:

- $\mathcal{N} = \{1, \ldots, n\}$ a finite set of *voters* (or *agents*).
- $\mathcal{X} = \{x_1, x_2, x_3, \ldots\}$ a set of alternatives/candidates/options.
- $\mathcal{L}(\mathcal{X})$ a set of linear orders on $\mathcal{X}$. Preferences/Ballots are elements of $\mathcal{L}(\mathcal{X})$.
- $x \succ_i y$: voters i prefers x to y.

A voting procedure is a function $F : \mathcal{L}(\mathcal{X})^n \rightarrow \mathcal{X}$

Voting rules

There are many different voting procedures, including these:

- Plurality: elect the candidate ranked first most often
- Single Transferable Vote (STV): keep eliminating the plurality loser until someone has an absolute majority
- Borda: each voter gives $m - 1$ points to the candidate they rank first, $m - 2$ to the candidate they rank second, etc.
- Copeland: award 1 point to a candidate for each pairwise majority contest won and 1 points for each draw
- Approval*: voters can approve of as many candidates as they wish, and the candidate with the most approvals wins

* does not fit in the previous formal definition.

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 maximising $s(x) = \sum_{i=1}^{n} score_i(x)$ (+ 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
d	d	b
a	c	c

plurality	borda	veto
a $\rightarrow$ 1	a $\rightarrow$ 6	a $\rightarrow$ 6
b $\rightarrow$ 0	b $\rightarrow$ 7	b $\rightarrow$ 6
c $\rightarrow$ 2	c $\rightarrow$ 6	c $\rightarrow$ 4
d $\rightarrow$ 1	d $\rightarrow$ 4	d $\rightarrow$ 4
c winner	b winner	a or b

Condorcet Winner

$N(x,y) = \#\{i \mid x \succ_i y\}$: 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	a**b**
d	c	b**a**
c	a	d

2 voters out of 3: $a \succ b$**b** $\succ$ **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: $d \succ c$

No Condorcet winnerb Condorcet winner Important note: no scoring rule is Condorcet-consistent.

Voting in Combinatorial Domain

Voting itself can be computationally difficult. Sometimes the domain have a combinatorial structure.

- Electing a committee of k members from amongst n candidates.
- During a referendum with more than one question

voters may be asked to vote on several propositions.

Example

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.

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

Basic Solution Attempts

- Solution 1: just vote for combinations directly
  - only feasible for very small problem instances
  - Example: 3-seat committee, 10 candidates = 120 options
- Solution 2: vote for top k combinations only (e.g., k = 1)
  - does address communication problem of Solution 1
  - possibly nobody gets more than one vote (tie-breaking decides)
- Solution 3: make a small preselection of combinations to vote on
  - does solve the computational problems
  - but who should select? (strategic control)

Combinatorial Vote

Idea: Ask voters to report their ballots by means of expressions in a **compact preference representation language** and apply your favourite voting procedure to the succinctly encoded ballots received.

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

Hint: compact representation language usually leads to increased complexity for some operations.

Preference Representation Languages

Several languages have been proposed for the compact representation of preference orders. See Lang (2004) for an overview. Examples:

- **Weighted goals**: use propositional logic to express goals; assign weights to express importance; aggregate (e.g., lexicographically)
- **CP-nets**: use a directed graph to express dependence between issues; use conditional preference tables to specify preferences on issue assuming those it depends on are fixed

All based on

- Set of variables $V = \{v_1, \ldots, v_m\}$
- Each variable v_i has a domain $D_i = \{d^i_1, \ldots, d^i_{k_i}\}$
- We will assume binary domains, so given vars $\{X, Y, Z\}$, a possible state/option is $\{x \neg yz\}$.

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

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

$$\begin{tikzcd} X \arrow[rr] & & Y \arrow[rr] & & Z \end{tikzcd}$$

$$\boxed{x \succ \neg x} \qquad \boxed{\begin{array}{c} x : y \succ \neg y \\ \neg x : \neg y \succ y \end{array}} \qquad \boxed{\begin{array}{c} x \vee y : z \succ \neg z \\ \neg x \wedge \neg z : \neg z \succ z \end{array}}$$

X independent of Y and Z; Y independent of Z.
$x : y \succ \neg y$ means that if $X = x$, then $Y = y$ is preferred to $Y = \neg y$, everything else being equal (ceteris paribus).

$$\begin{array}{cc} xyz \succ x\neg yz & xy\neg z \succ x\neg y\neg z \\ \neg x\neg yz \succ \neg xyz & \neg x\neg y\neg z \succ \neg xy\neg z \end{array}$$

Dominance problem

Question: *Does $a \succ b$ given the preference of an agent?*

- Trivial in classical setting with linear orders.
- What if the options are represented using CP-Nets?
  - Equivalent to finding a sequence of worsening flip sequence from a to b (or an improving one from b to a)
  - Example: $\{xyz\} \succ \{\neg x \neg y \neg z\}$?
    - $\{xyz\} \succ \{\neg xyz\} \succ \{\neg x \neg yz\} \succ \{\neg x \neg y \neg z\}$

$\boxed{x \succ \neg x}$

$$\begin{array}{|c|}\hline x : y \succ \neg y \\ \neg x : \neg y \succ y \\ \hline\end{array}$$

$$\begin{array}{|c|}\hline x \vee y : z \succ \neg z \\ \neg x \wedge \neg z : \neg z \succ z \\ \hline\end{array}$$

Hint: sequence of improving can be seen as sequence of actions = Planning

Classical Planning

Problem of finding a **sequence** of actions that **achieves a goal**, starting from a **given** initial state.
Expressed in **high-level** language

Example

- Init: p, q
- Goal: g
- Actions:
  - a Precondition: p. Effect: r
  - b Precondition: $q \wedge r$. Effect: g
  - c Precondition: q. Effect: $\neg q \wedge r$
- Plan: a, b

Classical Planning Syntax

Classical planning problems P are tuples of the form $P = \langle F, I, O, G \rangle$ where

- F: **fluent** symbols in the problem
- I: set of fluents true in the **initial situation**
- O: set of operators or actions. Every **action** a has
  - a **precondition** $Pre(a)$ given by a set of fluents
  - an **effect** $Eff(a)$ given by a set of fluents.
- G: set of fluents defining the **goal**

Classical Planning Model

- **Languages** such as Strips, ADL, PDDL, ..., represent **models** in compact form
- A **classical planner** is a **solver** over the class of **models** given by:
  - a **state space** S
  - a known **initial state** $s_0 \in S$
  - a set $S_G \subseteq S$ of **goal states**
  - **actions** $A(s) \subseteq A$ applicable in each $s \in S$
  - a deterministic **transition function** $s' = f(a, s)$ for $a \in A(s)$
  - uniform **action costs** $c(a, s) = 1$
- Given a problem $P = \langle F, I, O, G \rangle$, **states** of its corresponding model are **set of fluents** of P
- Their **solutions** (plans) are sequences of applicable actions that map s_0 into S_G

Planning (example)

Example

- Fluents: p, q, r, g
- Init: p, q
- Goal: g
- Actions:
  - a Precondition: p. Effect: r
  - b Precondition: $q \wedge r$. Effect: g
  - c Precondition: q. Effect: $\neg q \wedge r$

Model

- States: $\{p, q, r, g\}$, $\{p, q, r, \neg g\}$, $\{p, q, \neg r, g\}$, ...
- $s_0 = \{p, q, \neg r, \neg g\}$
- ...

CP-net dominance as planning

Idea: create a planning problem whose plan corresponds to an improving sequence.

- Given CP-net N with variables V, and a set of rules $C : y \succ \neg y$. Does $A \succ B$.
- Let P a classical planning problem with
  - Fluents: variables V
  - Init: B
  - Goal: A
  - For each rule $C : y \succ \neg y$, create action a with $Prec(a) = C \wedge \neg y$ and $Eff(a) = y$
- P has a plan iff $A \succ B$ in N.

Digression: What is planning anyway?

Planning is a form of symbolical reachability

(Enrico Giunchiglia. @UC3M - 2011)

- A plan is a particular sequence of transformations from a propositional state into other, using the language Strips and extensions.
- Typical criteria: length/cost of plan.
- CP-Net dominance does not depends on length of plan.

Towards combinatorial voting rules using CP-Nets

- Criteria:
  - Soundness
  - In relation with classical social choice theory
  - Effective
- What kind of voting rules are more promising?

  ... based in **Condorcet winner** notion, where a candidate x such that $\forall y \neq x, N(x, y) > \frac{n}{2}$
- Joint work with Ulle Endriss.

Multiagent unanimous dominance through planning

- Given a set of variables V, and n agents, each one with a CP-net N_i using variables V.
  Does $A \succ B$ for a majority of agents?.
- Let P a classical planning problem with
  - Fluents: v_i for each $v \in V$ and i agent
  - Init: B_i, for each agent i
  - Goal: A_i, for each agent i
  - For each rule $C : y \succ \neg y$ of agent i, create action a with $Prec(a) = C_i \wedge \neg y_i$ and $Eff(a) = y_i$
- P has a plan iff $A \succ B$ for **all** agents.

Multiagent unanimous dominance through planning

- Given a set of variables V, and n agents, each one with a CP-net N_i using variables V.
  Does $A \succ B$ for a majority of agents?.
- Let P a classical planning problem with
  - Fluents: v_i for each $v \in V$ and i agent
  - Init: B_i, for each agent i
  - Goal: A_i, for each agent i
  - For each rule $C : y \succ \neg y$ of agent i, create action a with $Prec(a) = C_i \wedge \neg y_i$ and $Eff(a) = y_i$
- P has a plan iff $A \succ B$ for **all** agents.

Multiagent dominance through planning

- *Does $A \succ B$ for a* **majority** *of agents?*.
- Let P a classical planning problem with
  - Fluents: v_i for each $v \in V$ and i agent. Also an atom d_i for each agent i.
  - Init: B_i, for each agent i. All d_i false.
  - Reward: $r(d_i) = 1$, $r(\neg d_i) = 0$
  - For each rule $C : y \succ \neg y$ of agent i, create action a with $Prec(a) = C_i \wedge \neg y_i$ and $Eff(a) = y_i$.
    Also for each agent i, action with precondition $Prec(a) = Y_i$, and $Eff(a) = d_i$
- Reward of a plan $a_1, \ldots, a_m$ with final state $s = \sum_{i=0}^{n} reward(d_i, s)$
- P has a plan with reward $> \frac{n}{2}$ iff $A \succ B$ for the majority of agents.

Multiagent dominance through planning (II)

- Formulation is equivalent to obtain a plan for $\frac{n}{2}$ separated.

- Size of problem: grows linear in number of agents.
  - Space-search grows exponential in size of planning problem.

Structure in Model-based AI

- SAT: unit propagation
- Bayesian Networks: variable elimination
- Planning: heuristics
- Constraints: ...
- ...

Sharing structure for multiagent dominance?

- Suppose the N voters can be divided in three groups A, B, C such that
  - Voters among a group have exactly the same CP-Net
  - Let's call them N_A, N_B, N_C.
- If $|A| + |B| > \frac{n}{2}$, then $A \succ B$ for the majority of agents.
- Two ideas
  - Plan for each group separated
  - Create a planning problem with variables labeled v_A, v_B, v_C.
- Further improvement by sharing variables in more general cases. *Work in progress!*

Towards combinatorial voting rules using partial orders

- Classical framework: linear orders
- CP-nets defined a partial order. It's probably the same for other languages.
- Redefine preferences for include indifference

Partial order preferences

Under what circumstances would we be willing to say that **the group of agents $\mathcal{N}$ prefers x to y**? Let's focus on having only two options: x and y.

- $\mathcal{P}(\{x, y\} := \{x \succ y\}, \{x \prec y\}, \{x \bowtie y\}$, the set of strict partial orders on x, y.

  $x \bowtie y$ is not indifference, but incomparability.
- A profile $R = (R_1, \ldots, R_n) \in \mathcal{P}(\{x, y\})^{\mathcal{N}}$ is a vector of such preferences order, one for each agent.
- An aggregation rule $F : \mathcal{P}(\{x, y\})^{\mathcal{N}} \rightarrow \mathcal{P}(\{x, y\})$ maps each each profile to a collective preference order.
- Let $\mathcal{N}^R_{x \succ y}$ denote the set of individuals who prefer x to y in R. Similarly for $\mathcal{N}^R_{x \prec y}$, $\mathcal{N}^R_{x \bowtie y}$.

Possible Definitions of Majority Dominance

- **Unanimous dominance**: given profile R, x dominates y under the unanimous rule if $\mathcal{N}^R_{x \succ y} = \mathcal{N}$.
- **Simple majority dominance**: given profile R, x dominates y under the unanimous rule if $|\mathcal{N}^R_{x \succ y}| > |\mathcal{N}^R_{x \prec y}|$
- **Absolute majority dominance**: given profile R, x dominates y under the unanimous rule if $|\mathcal{N}^R_{x \succ y}| > \frac{n}{2}$

Notice that **Absolute majority dominance** is useful deciding the Condorcet winner.

Axiomatic Analysis

- F is **anonymous** (A),
- F is **neutral** (N), if swapping x and y everywhere in the input profile results in a swap of x and y in the output.
- F satisfies **preference monotonicity** (PM), if additional support for a winner or diminished support for her opponent will never turn the former into a loser.
- F satisfies **uncertainty monotonicity** (UM), if an additional voter is undecided in an undecided election, the election will remain undecided.
- F satisfies **unanimous uncertainty elimination** (UUE), if for a previously undecided election all uncertain voters suddenly swing one way, then their will must be implemented.

Absolute Majority Rule

Theorem:An aggregation rule satisfies axioms (A), (N), (PM), (UM) and (UUE) iff it is the absolute majority rule.

Acknowledge

Most slides based on tutorials by:

- Ulle Endriss. Tutorial in Computational Social Choice. ECAI 2010.
- Jerome Lang. Computational aspects of voting. COST-ADT Doctoral School on Computational Social Choice 2010.

Summary

- Computational Social Choice study computational issues regarding collective decisions.
- Combinatorial Voting deals with domains where options are expressed in term of variables.
- From a classical point of view, Combinatorial Voting is not different than classical one, but very different from a computational point of view.
- CP-Nets is a language for representing the preferences of agents.
- Dominance between options can be formulated as a planning problem.
- Effective planning techniques could be useful for effective and practical Computational Social Choice.

References

- Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet. A Short Introduction to Computational Social Choice. Proc. SOFSEM-2007.
- Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet. Preference Handling in Com- binatorial Domains: From AI to Social Choice. AI Magazine, 29(4):37–46, 2008.
- J. Lang. Logical Preference Representation and Combinatorial Vote. Annals of Mathematics and Artificial Intelligence, 42(1–3):37–71, 2004.
- 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 AI Research, 21:135–191, 2004.
- Ronen Brafman. Tutorial on Preferences.