Computational Social Choice

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

Computational Social Choice

Social choice theory studies mechanisms for *collective decision making*, such as voting procedures or protocols for fair division.

Computational social choice adds a computational perspective to this, and also explores the use of concepts from social choice in computing.

- Part of wider trend of interdisciplinary research at the interface of *mathematical economics* (social choice, game and decision theory) and *computer science* (and artificial intelligence and logic);
- with an active research community, witness e.g. the COMSOC workshops in Amsterdam (2006) and Liverpool (2008).

Talk Outline

Part 1: Introduction to COMSOC by means of three examples —

- Applications of Computational Complexity to Voting Theory
- Variations of Classical Voting Theory for Applications in AI
- Logical Modelling and Formal Verification in Social Choice Theory

Part 2: Presentation of one subfield in more detail —

- Preference Modelling in Combinatorial Domains

Example from Voting

Suppose the *plurality rule* is used to decide an election: the candidate receiving the highest number of votes wins.

Assume the preferences of the people in, say, Florida are as follows:

$$
\begin{array}{ll}
49\%: & \text{Bush} \succ \text{Gore} \succ \text{Nader} \\
20\%: & \text{Gore} \succ \text{Nader} \succ \text{Bush} \\
20\%: & \text{Gore} \succ \text{Bush} \succ \text{Nader} \\
11\%: & \text{Nader} \succ \text{Gore} \succ \text{Bush}
\end{array}
$$

So even if nobody is cheating, Bush will win this election. But:

- In a *pairwise contest*, Gore would have defeated anyone.
- It would have been in the interest of the Nader supporters to *manipulate*, i.e., to misrepresent their preferences.

This is in fact a very general problem ...

Complexity as a Barrier against Manipulation

By the *Gibbard-Satterthwaite Theorem*, any voting rule for choosing between ≥ 3 candidates can be manipulated (unless it is dictatorial).

Idea: So it's always *possible* to manipulate, but maybe it's *difficult*! Tools from *complexity theory* can be used to make this idea precise.

- For the *plurality rule* this does *not* work: if I know all other ballots and want X to win, it is *easy* to compute my best strategy.
- But for *single transferable vote* it does work. Bartholdi and Orlin showed that manipulation of STV is *NP-complete*.

Recent work in COMSOC has expanded on this idea:

- NP is a worst-case notion. What about average complexity?
- Also: complexity of winner determination, control, bribery ...

J.J. Bartholdi III and J.B. Orlin. Single Transferable Vote Resists Strategic Voting. *Social Choice and Welfare*, 8(4):341–354, 1991.

Preferences and Ballots

Two common assumptions in voting theory:

- Voters have *preferences* that are *total orders* over candidates.
- Voters vote by submitting a structure just like their preferences, truthfully or not (*ballots* and preferences have *the same* structure).

We may want to drop these assumptions, because:

- For lack of information or processing resources, voters may be *unable to rank* all candidates (in their mind or on the ballot sheet).
- To reduce *complexity of communication*, we may want to design voting rules that work with ballots of bounded size.
- For *approval voting*, ballots cannot be encoded using total orders.

Variations of Classical Voting Theory

In recent work we have:

- Proposed a voting model where *preferences* and *ballots* can be different types of structures.
- Proposed a notion of *sincerity* to replace the standard notion of *truthfulness* (because the ballot language may *not allow* you to truthfully reproduce your preference).
- Obtained positive results for certain combinations:
  - Under *approval voting* with standard preferences, you can never benefit from not voting sincerely.
  - If you have *dichotomous preferences*, you can never benefit from not voting sincerely for a wide range of voting procedures.
  - Voting sincerely and effectively is *computationally tractable* in above scenarios.

U. Endriss, M.S. Pini, F. Rossi, and K.B. Venable. *Preference Aggregation over Restricted Ballot Languages: Sincerity and Strategy-Proofness*. Proc. IJCAI-2009.

Arrow's Impossibility Theorem

It seems reasonable to require a *social welfare function* (SWF), mapping profiles of individual preference orderings to a social preference ordering, to satisfy the following axioms:

- *Unanimity* (UN): if every individual prefers alternative x over alternative y, then so should society
- *Independence of Irrelevant Alternatives* (IIA): social preference of x over y should only depend on individual pref's over x and y
- *Non-Dictatorship* (ND): no single individual should be able to impose a social preference ordering

Theorem 1 (Arrow, 1951) *For more than two alternatives, there exists no SWF that satisfies all of (UN), (IIA) and (ND).*

K.J. Arrow. *Social Choice and Individual Values*. 2nd edition, Wiley, 1963.

Formal Verification of Arrow's Theorem

Logic has long been used to *formally specify* computer systems, facilitating formal or even *automatic verification* of various properties. Can we apply this methodology also to *social choice* mechanisms?

Tang and Lin (2009) show that the *"base case"* of Arrow's Theorem with 2 agents and 3 alternatives can be fully modelled in *propositional logic:*

- Automated theorem provers can verify $\textsc{Arrow}(2,3)$ to be correct in < 1 second — that's $(3!)^{3! \times 3!} \approx 10^{28}$ SWFs to check
- Opens up opportunities for quick sanity checks of hypotheses regarding new possibility and impossibility theorems.

Our ongoing work using *first-order logic* tries to go beyond such base cases.

P. Tang and F. Lin. Computer-aided Proofs of Arrow's and other Impossibility Theorems. *Artificial Intelligence*, 173(11):1041–1053, 2009

U. Grandi and U. Endriss. *First-Order Logic Formalisation of Arrow's Theorem*. Working Paper, University of Amsterdam, 2009.

Conclusion: Part 1

We have seen three examples for work in Computational Social Choice. Research can be broadly classified along two dimensions —

The kind of *social choice problem* studied, e.g.:

- aggregating individual judgements into a collective verdict
- electing a winner given individual preferences over candidates
- fairly dividing a cake given individual tastes

The kind *computational technique* employed, e.g.:

- algorithm design to implement complex mechanisms
- complexity theory to understand limitations
- logical modelling to fully formalise intuitions
- knowledge representation techniques to compactly model problems

Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet. *A Short Introduction to Computational Social Choice*. Proc. SOFSEM-2007.

Talk Overview: Part 2

Now for the main topic of the talk:

Preference Modelling in Combinatorial Domains

We will cover:

- What is the problem?
- Languages for compactly modelling preferences
- Examples for technical results
- Applications to collective decision making
- Conclusion

Social Choice in Combinatorial Domains

Many social choice problems have a *combinatorial structure:*

- Elect 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 n different propositions.
- Find a good *allocation* of n indivisible goods to agents.

Seemingly small problems generate huge numbers of alternatives:

- Number of 3-member committees from 10 candidates: $\binom{10}{3} = 120$ (i.e., $120! \approx 6.7 \times 10^{198}$ possible rankings)
- Allocating 10 goods to 5 agents: $5^{10} = 9765625$ allocations and $2^{10} = 1024$ bundles for each agent to think about

We need good *languages* for representing preferences!

Preference Representation Languages

The following are relevant questions to consider when we have to choose a preference representation language:

- *Cognitive relevance:* How close is a given language to the way in which humans would express their preferences?
- *Elicitation friendliness:* How difficult is it to elicit the preferences of an agent so as to represent them in the chosen language?
- *Expressive power:* Can the chosen language encode all the preference structures we are interested in?
- *Succinctness:* Is the representation of (typical) structures succinct? Is one language more succinct than the other?
- *Complexity:* What is the computational complexity of related problems, such as comparing two alternatives?

Combinatorial Domains

A *combinatorial domain* is a Cartesian product $\mathcal{D} = D_1 \times \cdots \times D_n$ of n finite domains. We want to represent *utility functions* over $\mathcal{D}$.

Typical cases are *allocation problems:* set $\mathcal{G}$ of indivisible goods; each agent has utility function $u : 2^{\mathcal{G}} \to \mathbb{R}$, mapping bundles to the reals.

That is, here each D_i is a *binary domain*, and $n = |\mathcal{G}|$.

Explicit Representation

The *explicit form* of representing a utility function u consists of a table listing for every bundle $S \subseteq \mathcal{G}$ the utility $u(S)$.

By convention, table entries with $u(S) = 0$ may be omitted.

- the explicit form is *fully expressive:*
  any utility function $u : 2^{\mathcal{G}} \rightarrow \mathbb{R}$ may be so described
- the explicit form is *not succinct:* it may require up to 2^n entries

Even very simple utility functions may require exponential space: e.g. the function $u : S \mapsto |S|$ mapping bundles to their cardinality.

Weighted Goals

A compact representation language for modelling utility functions over products of binary domains —

Notation: finite set of propositional letters PS; propositional language $\mathcal{L}_{PS}$ over PS to describe requirements, e.g.:

$$p, \quad \neg p, \quad p \wedge q, \quad p \vee q$$

A *goalbase* is a set $G = \{(\varphi_i, \alpha_i)\}_i$ of pairs, each consisting of a (consistent) propositional formula $\varphi_i \in \mathcal{L}_{PS}$ and a real number α_i. The utility function u_G generated by G is defined by

$$u_G(M) \quad = \quad \sum \{\alpha_i \mid (\varphi_i, \alpha_i) \in G \text{ and } M \models \varphi_i\}$$

for all models $M \in 2^{PS}$. G is called the *generator* of u_G.

Example: $\{(p \vee q \vee r, 7), (p \wedge q, -2), (\neg s, 1)\}$

A Family of Languages

By imposing different restrictions on formulas and/or weights we can design different representation languages.

Regarding *formulas*, we may consider restrictions such as:

- *positive* formulas (no occurrence of $\neg$)
- *clauses* and *cubes* (disjunctions and conjunctions of literals)
- k*-formulas* (formulas of length $\leq k$), e.g. 1-formulas = literals
- combinations of the above, e.g. k-pcubes

Regarding *weights*, interesting restrictions would be $\mathbb{R}^+$ or $\{0, 1\}$.

If $H \subseteq \mathcal{L}_{PS}$ is a restriction on formulas and $H' \subseteq \mathbb{R}$ a restriction on weights, then $\mathcal{L}(H, H')$ is the language conforming to H and H'.

Properties

We are interested in the following types of questions:

- Are there restrictions on goalbases such that the utility functions they generate enjoy natural structural properties?
- Are some goalbase languages more succinct than others?
- What is the complexity of reasoning about preferences expressed in a given language?

J. Uckelman, Y. Chevaleyre, U. Endriss, and J. Lang. Representing Utility Functions via Weighted Goals. *Mathematical Logic Quarterly*, 55(4):341–361, 2009.

Expressive Power

An example for a language that is fully expressive:

Theorem 2 (Expressivity of pcubes) *$\mathcal{L}(pcubes, \mathbb{R})$, the language of positive cubes, can express all utility functions.*

Proof sketch: Show how to build a goalbase for any given function u: (1) $\top$ must get weight $u(\emptyset)$. (2) Weights of longer formulas are uniquely determined by the weights of their subformulas. ✓

In fact, $\mathcal{L}(pcubes, \mathbb{R})$ has a *unique* way of representing any given u.

$\mathcal{L}(cubes, \mathbb{R})$, for example, is also fully expressive, but not unique:
$\{(p \wedge q, 5), (p \wedge \neg q, 5), (\neg p \wedge q, 3), (\neg p \wedge \neg q, 3)\} \equiv \{(\top, 3), (p, 2)\}$

Expressive Power: Modular Functions

A function $u : 2^{PS} \to \mathbb{R}$ is *modular* if for all $M_1, M_2 \subseteq 2^{PS}$ we have:

$$u(M_1 \cup M_2) = u(M_1) + u(M_2) - u(M_1 \cap M_2)$$

Here's a nice characterisation of the modular functions:

Theorem 3 (Expressivity of literals) *$\mathcal{L}(literals, \mathbb{R})$, the language of literals, can express all modular utility functions, and only those.*

Relative Succinctness

If two languages can express the same class of utility functions, which should we use? An important criterion is *succinctness*.

Let $\mathcal{L}$ and $\mathcal{L}'$ be two languages that can define all utility functions belonging to some class $\mathcal{U}$.

We say that *$\mathcal{L}'$ is at least as succinct as $\mathcal{L}$* over $\mathcal{U}$ if there exist a mapping $f : \mathcal{L} \rightarrow \mathcal{L}'$ and a *polynomial* function p such that for all expressions $G \in \mathcal{L}$ with the corresponding function $u_G \in \mathcal{U}$:

- $u_G = u_{f(G)}$ (they both represent the same function); and
- $size(f(G)) \leq p(size(G))$ (polysize reduction).

Explicit Form vs Positive Cubes

Both the explicit form and $\mathcal{L}(pcubes, \mathbb{R})$, the language of positive cubes, are fully expressive. Which is more succinct?

Theorem 4 (Explicit form and positve cubes) *The explicit form and $\mathcal{L}(pcubes, \mathbb{R})$ are incomparable in terms of succinctness.*

Proof: These functions prove the mutual lack of a polysize reduction:

- $u_1(X) = |X|$: requires n weighted 1-pcubes (*linear*); but $2^n - 1$ non-zero values in the explicit form (*exponential*). ✓
- $u_2(X) = 1$ for $|X| = 1$ and $u_2(X) = 0$ otherwise: requires n non-zero values in the explicit form (*linear*); but $2^n - 1$ pcubes (*exponential*) — all cubes of length k need weight $(-1)^{k+1} \times k$. ✓

But: *interesting* functions usually more succinct in $\mathcal{L}(pcubes, \mathbb{R})$

The Efficiency of Negation

If we allow *negation* in our language, we can do better than either one of the two languages considered before:

Theorem 5 (Cubes and pcubes) *The language $\mathcal{L}(cubes, \mathbb{R})$ is strictly more succinct than the language $\mathcal{L}(pcubes, \mathbb{R})$.*

Theorem 6 (Cubes and explicit form) *The language $\mathcal{L}(cubes, \mathbb{R})$ is strictly more succinct than the explicit form.*

Computational Complexity

Other interesting questions concern the complexity of reasoning about preferences. Consider the following decision problem:

> $\textsc{MaxUtil}(H, H')$
> **Instance:** Goalbase $G \in \mathcal{L}(H, H')$ and $K \in \mathbb{Z}$
> **Question:** Is there an $M \in 2^{PS}$ such that $u_G(M) \geq K$?

Complexity results include:

- $\textsc{MaxUtil}(\mathit{forms}, \mathbb{R})$ is *NP-complete* (and not worse).
- Even $\textsc{MaxUtil}(\mathit{2\text{-}pcubes}, \mathbb{R})$ is *NP-complete*.
- But $\textsc{MaxUtil}(\mathit{pforms}, \mathbb{R}^+)$ and $\textsc{MaxUtil}(\mathit{literals}, \mathbb{R})$ are *easy*.

Also interesting: What is the complexity of finding an allocation that maximises *utilitarian* or *egalitarian social welfare* (for language X)?

Application: Distributed Negotiation

Scenario: indivisible goods; agents with valuation functions

Goal: Want to design negotiation protocols for agents with good properties, ideally fast convergence to a socially optimal state.

Preference representation is one of several parameters in the model. Explicit modelling of the language has several advantages:

- Can guide *elicitation* of preferences from agents.
- Can characterise special *classes of preferences* that avoid impossibilities, allow for simpler protocols, etc.
- Permits *complexity* analysis.

Y. Chevaleyre, U. Endriss, S. Estivie, and N. Maudet. Multiagent Resource Allocation in k-additive Domains: Preference Representation and Complexity. *Annals of Operations Research*, 163(1):49–62, 2008.

Application: Combinatorial Auctions

Combinatorial Auction: auction for simultaneously selling several items (with complements and substitutes)

Example: CA for a pair of shoes *vs* two auctions for one shoe each

Bidding is the process of communicating one's preferences to the auctioneer (truthfully or otherwise). Can use *goalbase languages!*

Winner determination is the problem faced by the auctioneer to decide which goods to award to which bidder.

- Winner determination is known to be *NP-hard*.
- *Heuristic-guided search* (AI technique) can often give optimal solution in reasonable time.

J. Uckelman and U. Endriss. *Winner Determination in Combinatorial Auctions with Logic-based Bidding Languages*. Proc. AAMAS-2008.

Conclusion

- Combinatorial explosion ⇒ number of alternatives can get huge
  ⇒ collective choice mechanisms need to be adapted
- Logic-based languages are good candidates for modelling preferences in combinatorial domains.
- Wider research area: Computational Social Choice
- Papers are on my website (including the surveys below):

http://www.illc.uva.nl/~ulle/

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 Combinatorial Domains: From AI to Social Choice. *AI Magazine*, Winter 2008.