Computational Social Choice Theory: Winner Determination and Manipulation

Konstantinos Bampalis

Birkbeck College, University of London, School of Computing and Mathematical Sciences

October 2024

Outline

- Voting and Computational Complexity
- Computational Social Choice Theory
- Problems
  - Winner Determination
  - Manipulation
- Focus on the Manipulation Problem

Voting

- Alice, Bob and Charlie want to go on holiday together
- Need to choose a place among three alternatives; Athens, Rome, Barcelona
- Different preferences; e.g. Alice: Athens $>$ Barcelona $>$ Rome
- How to pick a destination?
- Easy! By voting

Voting

- Before they vote, they need to choose a voting system.
- A mechanism that will aggregate individual preferences into a collective outcome
- Many different voting systems exist!
- Social Choice Theory studies different collective decision procedures and their properties.

Definition

Let N be a group of voters and A be a set of alternatives. A voter is represented by their preferences over A, where P_i denotes the preferences of voter i. Let $\mathcal{L}^n$ denote the set of all possible preferences over A. A Preference Profile is a set $P \subseteq \mathcal{L}^n$, where $P_i \in P$ for all $i \in N$

Voting

Definition

A Voting Mechanism is a rule that dictates how to pick the winner of an election. That is, a map $\mathcal{E} : \mathcal{L}^n \to 2^A$.

Voting

- Many different types of Voting Rules
- Based on the following ideas:
  - how many points a candidate collects
  - Head-to-Head Contests between candidates

Voting

- Scoring Rules: a scoring vector α determines how many points a candidate obtains
- The candidate with the most points wins
- Different rules based on different ways to allocate points
- Plurality, k-Approval, Borda

Voting

- Pairwise Comparisons: Candidates do not receive points immediately.
- Instead, all candidates engage in a head-to-head contest
- And then check the winner according to some majority criterion
- Condorcet, Copeland, Maximin

Voting

- A Voting Rule is nothing more than a function
- There is no correct system
- Look for various criteria our systems want to satisfy and combinations thereof
- Many Impossibility Results

Voting

- Some properties we look at:
  1. Non-Dictatorship
  2. Resoluteness
  3. Sovereignty
  4. Strategy Proofness

Theorem (Gibbard-Satterthwaite)

If there are at least 3 candidates, there is no preference-based voting system that simultaneously satisfies Non-Dictatorship, Resoluteness, Sovereignty and Strategy-Proofness.

Voting

- Most voting systems are not dictatorial, lead to a single candidate being elected and do not exclude any candidate from the outset.
- All voting Rules are manipulable

Computational Complexity Theory

- Computational Complexity Theory studies:
  - i studies how hard it is to solve a problem
  - ii classifies problems into classes of similar difficulty
  - iii studies the relationships between these class
- What is a problem?
- How do we solve problems?
- What do we mean by classifying and by difficulty?

Computational Complexity Theory

- A Problem is a question that we would like to answer for some input, X. For example:
  i. Input Graph G; Is G 3-Colourable?
  ii. Input Formula ϕ Is ϕ satisfiable?
- Focus on Decision Problems
- Given input x and some property P, does x satisfy P?

Computational Complexity Theory

Definition (Alphabet)

An alphabet Σ is a finite set of symbols. A string w over Σ is a finite sequence. The set Σ^* denotes all strings over Σ.

Definition (Language)

A language $\mathcal{L}$ is a subset of Σ^*.

Definition (Boolean Function)

Let $\Sigma = \{0, 1\}$. A Boolean Function is $f : \Sigma^* \to \Sigma$.

Computational Complexity Theory

- We use the *Binary Alphabet*, $\Sigma = \{0, 1\}$
- All inputs (finite objects) are encoded over Σ^*
- All outputs are encoded over Σ

Computational Complexity Theory

- Decision Problems
- the set of inputs for which the answer is yes;
- $\mathcal{L}_f := \{x : f(x) = 1\}$

Computational Complexity Theory

- We solve problems using *Algorithms*
- A set of fixed rules that can be carried out mechanically
- Given each input to decide membership in the language
- A model of computation specifies these rules:
- Turing Machine (Algorithm)

Computational Complexity Theory

- Hardness is measured by looking at the computational resources required to solve a problem
- Time (and Space)
- Time Complexity is measured by counting the number of steps an algorithm takes to solve a problem

Computational Complexity Theory

- Problems of similar levels of difficulty are placed into sets
- Complexity Classes
  - The class $\mathbf{P}$: easy to solve
  - The class $\mathbf{NP}$: easy to verify - hard to solve (?)
  - and many other classes...
- Relationships between classes: Is $\mathbf{P} \subseteq \mathbf{NP}$?

Computational Complexity Theory

Definition

The class $\mathbf{P}$ is the set of languages decidable in polynomial time by a Turing Machine.

Definition

The class $\mathbf{NP}$ is the set of languages whose solution can be verified in polynomial time.

Computational Complexity Theory

- Problems X and Y are the same if we can convert X to Y, and conversely
- We say that X is *Polynomially Reducible* to Y

Computational Complexity Theory

- Within $\mathbf{NP}$ there are many problems that are basically the same problem
- $\mathbf{NP}-$Complete
  i in $\mathbf{NP}$
  ii reducible to all other problems in $\mathbf{NP}$.
- A polynomial algorithm for one implies a polynomial algorithm for all, and conversely.
- To prove that a problem is hard; Reduce to known $\mathbf{NP}-$Complete problem

Computational Complexity Theory

- Let ϕ be a propositional formula in CNF
- Determine if ϕ is satisfiable
- $\texttt{SAT}$ is $\mathbf{NP}-$Complete

Computational Social Choice Theory

- COMSOC is a unique field that combines Mathematics, Theoretical Computer Science and Social Choice Theory, among others.
- From Social Choice Theory to Computer Science:
  - Social Theoretic ideas applied to areas in CS where elfish software agents have competing preferences. (AI, ranking algorithms, etc.)
- From Computer Science to Social Choice Theory:
  - Complexity Theoretic ideas applied to deal with Computational and Algorithmic aspects of Social Choice Theory.

Problems in Computational Social Choice Theory

- Focus: From Computer Science to Social Choice Theory
- Pick any Voting Mechanism, $\mathcal{E}$
  - `WINNER`: Can $\mathcal{E}$ determine a winner quickly?
  - `MANIPULATION`: Can $\mathcal{E}$ be easily manipulated?
- We want an $\mathcal{E}$ that determines a winner *Easily*.
- And that is *Hard* to manipulate.

The Winner Problem

- The outcome of the election needs to be determined quickly
- Let $\mathcal{E}$ be any mechanism, based on the scoring rule
- Then $\mathcal{E}$-WINNER $\in \mathbf{P}$
  - Why? Adding points is easy!
- Not always the case
  - Exploration of such a rule in dissertation

The Manipulation Problem

- All Voting Rules are susceptible to *Manipulation*
- (Computational) Hardness is good! If it is hard for a manipulator to manipulate an election then we are protected
- Computational Complexity allows us to get around this problem

The Manipulation Problem

$|N|$ voters, $|A|$ candidates, voting rule $\mathcal{E}$, manipulator i and preferences profiles for all $N - \{i\}$

Let p be a distinguished candidate

Can i construct a preference profile that ensures p wins?

If the question can be answered quickly, then Bad News!

Let's see some results!

Easy Manipulation

- `GreedyManipulation`
- Place p at the top of P_i
- While there are unranked candidates from A
  - If there exists $x \in A$ such that x does not prevent p from winning then place him in the next position
  - else output $P_i = \emptyset$

Easy Manipulation

- $A = \{a, b, c, p\}$, $N = \{1, 2, 3\}$ under Borda Count
- $P_1 = P_2 : a > p > b > c$
- Construct P_3 such that p wins

Easy Manipulation

- Begin by ranking p first
- p has 7 points.
- a has 6 points and hence can't go in the second place
- Put b second.
- So far $P_3 : p > b$
- a still can't go in the third place, put c
- Finally: $P_3 : p > b > c > a$

Easy Manipulation

- `GreedyManipulation` is Correct
  - Why? Complicated, the proof is in the dissertation
- `GreedyManipulation` is Efficient
  - Why? The while loop will run at most $|A| - 1$ times.

Theorem

GreedyManipulation will construct Pi that makes p the winner, or conclude that such ordering does not exist, for any voting scheme $\mathcal{E}$ out of: Plurality, Borda, Maximin or Copeland

Corollary

$\mathcal{E}-$Manipulation $\in \mathbf{P}$ for the above Voting Rules.

From Easy to Hard Manipulation

- Good news!
- Modify existing mechanisms to make manipulation hard
  - Pre-Rounds: add a simple round that eliminates candidates, then run the existing mechanism
  - Hybrid Rules: combine two rules together - eliminate candidates using the first rule, then run the other rule on the remaining ones.
- These simple tweaks can make manipulation hard for all the voting rules mentioned above.