Computational Social Choice and Complexity Theory

Ronald de Haan

University of Amsterdam

https://staff.science.uva.nl/r.dehaan/esslli2018
me@ronalddehaan.eu

ESSLLI 2018 – **Day 2**

Recap

- Voting theory, SCFs, SWFs
- Plurality, Borda, IRV, Kemeny
- Computational complexity theory
- Complexity of the Winner Determination problem for different voting rules

Online Tool to Compute Winner Determination

P	3x	3x	4x	4x	1x	1x
#1	b	d	b	a	a	e
#2	d	a	c	e	b	c
#3	a	c	e	c	c	d
#4	e	e	d	d	e	a
#5	c	b	a	b	d	b

```
5
1,a
2,b
3,c
4,d
5,e
16,16,6
3,2,4,1,5,3
3,4,1,3,5,2
4,2,3,5,4,1
4,1,5,3,4,2
1,1,2,3,5,4
1,5,3,4,1,2
```

- http://democratix.dbai.tuwien.ac.at/

Online Computational Social Choice Tool

- Spliddit: a tool to fairly divide rent, (housework) tasks, taxi fare, course credit, etc.
- `http://www.spliddit.org/`

What we'll do today

- Parameterized complexity theory
- Revisit the complexity of Winner Determination
- Strategic manipulation in voting
- The "computational defense" against manipulation, and its caveats

Winner Determination for the Kemeny rule

- Recap: Winner Determination for the Kemeny rule is computationally intractable (Θ_2^p-complete)
- Does this mean that we cannot use this voting rule in any situation?
- Not quite..
- To see why, we need to have a look at parameterized complexity theory

Parameterized Complexity Theory

Example: Graph 3-Coloring

- Graph 3-Coloring is NP-complete
- *Meaning:* in the worst case, we need to spend a lot of time to solve the problem (assuming P $\neq$ NP)
- Extreme and ridiculous example:
  - What if for our application we only have (large) graphs with very few edges, say, ≤ 10 (most nodes are isolated)
  - Does the worst case NP-hardness result say much about this setting?
  - **No**, we can solve the problem in time $O(3^{10} n^c) = O(n^c)$

Parameterized complexity

- Traditionally, complexity theory measures running time only in terms of the size n of the input
- Because it measures worst-case complexity, results are overly negative
- Parameterized Complexity Theory:
  - *Main idea:* you know more about the structure of the inputs for your problem than just the size in bits
    $\Rightarrow$ take this knowledge into account
  - Measure running times in terms of input size n and a parameter k
  - The parameter captures structure that is present in the input
  - Fixed-parameter tractability: running time of $f(k) \cdot n^c$, for some (computable) function f and some constant c

Parameterized Complexity Classes

- Parameterized Decision Problem:
  - A formal language $Q \subseteq \Sigma^* \times \mathbb{N}$
  - Each input $(x, k) \in \Sigma^* \times \mathbb{N}$ consists of the main input x and the parameter k
- FPT:
  - All problems $Q \subseteq \Sigma^* \times \mathbb{N}$ that can be solved in time $f(k) \cdot n^c$ for some computable function f (possibly exponential or worse) and some constant c
- W[1], W[2], . . . , para-NP:
  - Parameterized intractability classes
  - Idea: parameterized analogues of NP

Parameterized Reductions

- To give evidence that a problem is not fixed-parameter tractable, we use the notion of hardness
- A parameterized reduction (or fpt-reduction) from one parameterized problem $Q_1 \subseteq \Sigma^* \times \mathbb{N}$ to another parameterized problem $Q_2 \subseteq \Sigma^* \times \mathbb{N}$ is a function $f : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ such that:
  - for each $(x, k) \in \Sigma^* \times \mathbb{N}$, it holds that $(x, k) \in Q_1$ if and only if $f(x, k) = (x', k') \in Q_2$,
  - $f(x, k)$ is computable in time $h(k) \cdot |x|^c$, for some computable function h and some constant c
  - $k' \leq g(k)$ for some computable function g

Parameterized Hardness

- Let K be a parameterized complexity class, e.g., $K = \mathrm{W}[1]$.
- A parameterized problem $Q \subseteq \Sigma^* \times \mathbb{N}$ is K-hard if for all problems $Q' \in \mathrm{K}$ there is a parameterized reduction from Q' to Q
  - If Q is fixed-parameter tractable, then all problems in K are fixed-parameter tractable!
- A parameterized problem is K-complete if it is both K-hard and in K
- The classes W[1], W[2], ..., para-NP are widely believed (but not proven) to be different from FPT

  Under the assumption that this is true: W[1]-hard $\Rightarrow$ not FPT

Example: Graph c-Coloring

- The input is an undirect graph
  - A finite set N of nodes
  - A finite set E of edges $\{n_1, n_2\}$ with $n_1, n_2 \in N$
- The task is to decide if you can color each node with a color in $\{1, 2, \dots, c\}$ so that no two connected nodes have the same color
- c-Coloring is NP-complete
- Let's look at some parameterized versions of this problem..

Example: Graph c-Coloring

- Parameter $k = c$
  - para-NP-complete
  - *Idea:* for $k = 3$ the problem is NP-complete

    an fpt-algorithm would give a polynomial-time algorithm for $k = 3$
- Parameter $k = \#$ of vertices of degree > 2
  - fixed-parameter tractable:
  - *Idea:* try out all possible colorings for the vertices of high degree (at most k^k of them)

    for each of these, extend them to a coloring to all vertices

Kemeny voting

- Winner Determination for the Kemeny rule is NP-hard (Θ_2^p-complete)
- What if we take as parameter $k = |A|$: the # of candidates?
  - Reasonable: many elections have, say, ≤ 10 candidates

Theorem

Winner Determination for the Kemeny voting rule is fixed-parameter tractable for $k = |A|$.

- *Proof (idea):*
  - there are $k!$ preference orders
  - iterate over all of them, and for each compute the Kendall-Tau distance to the profile P
  - select the ones with smallest distance, and take their top-ranked candidates as winners
  - this runs in time $O(k! \cdot n^c)$ for some constant c

Kemeny voting

Theorem (Betzler, Fellows, Guo, Niedermeier, Rosamond, 2009)

Winner Determination for the Kemeny voting rule is solvable in time $O(2^k k^2 n)$ for $k = |A|$.

- So the Kemeny voting rule can be used with reasonable efficiency for up to, say, 20 candidates

N. Betzler, M.R. Fellows, J. Guo, R. Niedermeier, and F.A. Rosamond. Fixed-parameter algorithms for Kemeny rankings. Theoretical Computer Science, 410(45), 4554-4570, 2009.

Strategic Manipulation

Strategic Manipulation

P	1	2	3	4	5	6	7	8	9
#1	a	a	b	b	b	b	d	d	d
#2	b	b	a	a	c	c	c	c	c
#3	c	c	c	c	a	a	a	a	a
#4	d	d	d	d	d	d	b	b	b

- You are voter 9
- We use Borda voting
- Do you want to change your vote?
- What are the current Borda scores?

  a: 15, b: 16, c: 14, d: 8

Strategic Manipulation

P	1	2	3	4	5	6	7	8	$9'$
#1	a	a	b	b	b	b	d	d	a
#2	b	b	a	a	c	c	c	c	d
#3	c	c	c	c	a	a	a	a	c
#4	d	d	d	d	d	d	b	b	b

- You are voter 9
- We use Borda voting
- Do you want to change your vote?
- What are the current Borda scores?

  a: 15, b: 16, c: 14, d: 8
- What are the new Borda scores?

  a: 17, b: 16, c: 13, d: 7

Strategic Manipulation

- Let F be a voting rule (an SCF) and let $P = (\succ_1, \ldots, \succ_n)$ be a profile

  (for some set N of voters and some set A of alternatives)

- A strategic manipulation for agent $i \in N$ consists of an insincere preference order $\succ_i' \neq \succ_i$ such that:

$$F(P') \succ_i F(P),$$

  where:

$$P' = (\succ_1, \ldots, \succ_{i-1}, \succ_i', \succ_{i+1}, \ldots, \succ_n) = (P_{-i}, \succ_i')$$

- That is, a unilateral deviation from the truth by some agent that gives her a more preferred outcome

The Gibbard-Satterthwaite Theorem

- Can we find voting rules that are not manipulable?
- A rule that is not manipulable is called strategyproof

Theorem (Gibbard, 1973; Satterthwaite, 1975)

Any resolute, nonimposed, and strategyproof SCF for three or more alternatives is a dictatorship.

- resolute: for every profile there is exactly one winner
- nonimposed: every alternative wins in at least one profile
- dictatorship: there is one voter (the dictator) whose top candidate is the winner, no matter what the profile is

Limitations of the Gibbard-Satterthwaite Theorem

- Can we get around the Gibbard-Satterthwaite Theorem?
  - Limitation 1: the manipulator needs to know the ballot of the other voters
  - Limitation 2: the manipulator needs to be sure that no other voter will change her mind
  - Limitation 3: the result only applies to resolute voting rules
  - Limitation 4: the manipulator needs to have the computational resources to predict whether a change in her ballot will change the outcome in her favor

The "Computational Defense"

- The defense:
  - If the following computational problem is intractable:

Manipulation(F)

Input: a set N of voters, a set A of candidates, a preference profile P (for N and A), and a candidate $a^* \in A$

Output: Is there some $\succ_i'$ such that $a^* \in F(P_{-i}, \succ_i')$?

then no voter can effectively manipulate the election
(the rule is resistant to manipulation)

J.J. Bartholdi III, C.A. Tovey, and M.A. Trick. The Computational Difficulty of Manipulating an Election. Social Choice and Welfare, 6(3):227–241, 1989.

Strategic Manipulation for Kemeny Voting

P	1	2	3	4	5	6	7	8	9
#1	a	a	b	b	b	b	d	d	d
#2	b	b	a	a	c	c	c	c	c
#3	c	c	c	c	a	a	a	a	a
#4	d	d	d	d	d	d	b	b	b

- Again, you are voter 9
- We use Kemeny voting
- Do you want to change your vote?
- Current Kemeny winners: a, b

Strategic Manipulation for Kemeny Voting

P	1	2	3	4	5	6	7	8	$9'$
#1	a	a	b	b	b	b	d	d	a
#2	b	b	a	a	c	c	c	c	d
#3	c	c	c	c	a	a	a	a	c
#4	d	d	d	d	d	d	b	b	b

- Again, you are voter 9
- We use Kemeny voting
- Do you want to change your vote?
- Current Kemeny winners: a, b
- New Kemeny winners: a

Manipulating Kemeny and IRV

- Both for the Kemeny rule and for IRV, manipulation is intractable

Theorem

Manipulation(Kemeny) *is Θ_2^p-complete.*

Theorem (Bartholdi, Orlin, 1991)

Manipulation(IRV) *is NP-complete.*

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

Manipulating Borda and Plurality

- A voting rule F satisfies the BTT conditions if:
  - Winner determination for F is polynomial-time solvable
  - For every profile P and every alternative a, the rule assigns a score $S(P, a)$ to a
  - For every profile P, the alternative with the highest score wins
  - The following monotonicity property holds:
    for any two profiles P, P' and any alternative a:
    if for each voter i it holds that $\{\, b : a \succ_i b \,\} \subseteq \{\, b : a \succ'_i b \,\}$,
    then $S(P, a) \leq S(P', a)$.

Theorem (Bartholdi, Tovey, Trick, 1989)

If F satisfies the BTT conditions, then Manipulation(F) *is polynomial-time solvable.*

- So Manipulation(Borda) and Manipulation(Plurality) are polynomial-time solvable

Limits of the "Computational Defense"

- Computational hardness results are typically worst case results
- This means that there is no algorithm that decides whether (and how) to manipulate and that is efficient for all profiles
- However, there might be algorithms that are efficient for a subset of profiles

Manipulating Kemeny and IRV (Revisited)

- Suppose we only have a small number of alternatives, say, $|A| = 10$

Theorem

Manipulation(Kemeny) *and* Manipulation(IRV) *are fixed-parameter tractable for* $k = |A|$.

- *Proof (idea):*
  - compute the current winners
  - there are $k!$ preference orders, iterate over all of them, and for each compute the new winner(s)
  - if there is a new set of winners that is preferred over the current winners, submit the corresponding preference order
  - this can be done in time $O(k!\,2^k k^2 n)$ for Kemeny and in time $O(k!\,n^c)$ for IRV

Variants of Manipulation

- Individual vs. group / coalitional manipulation: a single vs. multiple agents manipulating cooperatively
- Weighted manipulation: the votes in the election have different weights
- Constructive / destructive manipulation: the aim is to make one alternative $a^* \in A$ win / lose
- Optimistic / pessimistic manipulation: count the tie-breaking in favor of / against the manipulators
- Incomplete information: the manipulator(s) have uncertainty about the votes in the profile
- Higher-order manipulation: take into account other manipulators' (counter) moves
- etc.

Hardness for Coalitional Weighted Manipulation for Veto

- Veto is the voting rule that, for each voter, gives the top $|A| - 1$ candidates on that voters' ballot one point each, and selects as winners the candidates with highest score.
- Coalitional Weighted Manipulation:
  - Coalitional: there is a coalition of voters that together aim to make one candidate $a^* \in A$ win
  - Weighted: each voter $i \in N$ is associated with a weight $w_i \in \mathbb{N}$

Theorem (Conitzer, Sandholm, Lang, 2007)

The coalitional weighted manipulation problem for the Veto rule is NP-complete.

V. Conitzer, T. Sandholm, and J. Lang. When Are Elections with Few Candidates Hard to Manipulate? Journal of the ACM, 54(3), 1–33, 2007.

Hardness for Coalitional Weighted Manipulation for Veto

- Partition: given integers $w_1, \ldots, w_m$ with $\sum_{1 \leq i \leq m} w_i = W$, is there a subset $S \subseteq \{1, \ldots, m\}$ such that $\sum_{i \in S} w_i = W/2$?
- A reduction from Partition to CWM for Veto:
  - Let $A = \{a, b, c\}$, and let $a = a^*$
  - Create one non-manipulator voter 0 with weight $W - 1$ that ranks a last (e.g., $b \succ c \succ a$)
  - For each $i \in \{1, \ldots, m\}$, create a manipulator voter i with weight $2w_i$
- *Idea*: the manipulators can make a win if and only if there exists some $S \subseteq \{1, \ldots, m\}$ such that $\sum_{i \in S} w_i = W/2$
  - All voters $i \in S$ rank b last, and all voters $j \in \{1, \ldots, m\} \setminus S$ rank c last, giving a the score of $2W$ and b and c the score of $2W - 1$

Recap

- Parameterized complexity theory
- Revisit the complexity of Winner Determination
- Strategic manipulation in voting
- The "computational defense" against manipulation, and its caveats

Homework exercise

- Sketch a proof of the following theorem:

Theorem (Bartholdi, Tovey, Trick, 1989)

If F satisfies the BTT conditions, then Manipulation(F) *is polynomial-time solvable.*

- (Hint: look at Section 6.4 of the Handbook of Computational Social Choice.)

F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. Procaccia. Handbook of Computational Social Choice. Cambridge University Press, 2016.

Spoiler!

(Solution for homework exercise)

Manipulating with the BTT Conditions

Theorem (Bartholdi, Tovey, Trick, 1989)

If F satisfies the BTT conditions, then Manipulation(F) *is polynomial-time solvable.*

- *Idea:*
  - Rank a^* first
  - Repeat:
    - Find some alternative b to rank next so that b does not get a higher score than a^*

      (to check whether some b has this property: rank b next, arbitrarily complete the ranking, and compute the scores

      by the monotonicity property this tells you what you need to know about the scores of b and a^*)
    - If you can't find such a b, fail!
    - If you find a complete ranking, success!

BTT Conditions (Repeated)

- A voting rule F satisfies the BTT conditions if:
  - Winner determination for F is polynomial-time solvable
  - For every profile P and every alternative a, the rule assigns a score $S(P, a)$ to a
  - For every profile P, the alternative with the highest score wins
  - The following monotonicity property holds:

    for any two profiles P, P' and any alternative a:
    if for each voter i it holds that $\{ b : a \succ_i b \} \subseteq \{ b : a \succ'_i b \}$,
    then $S(P, a) \leq S(P', a)$.