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 1**

Information

- Course website:

  https://staff.science.uva.nl/r.dehaan/esslli2018/

  (also linked on the main ESSLLI website)

  - The slides will be available
  - Pointers to additional reading material

- About me:
  - I studied linguistics, cognitive artificial intelligence, computational logic, (parameterized) complexity theory
  - Over the last few years, I started researching computational social choice
  - This is my 10th ESSLLI :-)

First Point

- If you have a question at any point, please ask!
  - If I don't want to answer it, I will tell you. ;-)

Computational Social Choice

"Computational social choice is an interdisciplinary field of study at the interface of social choice theory and computer science, promoting an exchange of ideas in both directions."

http://research.illc.uva.nl/COMSOC/what-is-comsoc.html

- Computational social choice topics:
  - voting protocols
  - resource allocation and fair division algorithms
  - stable matching
  - coalition formation
  - judgment aggregation
  - ...

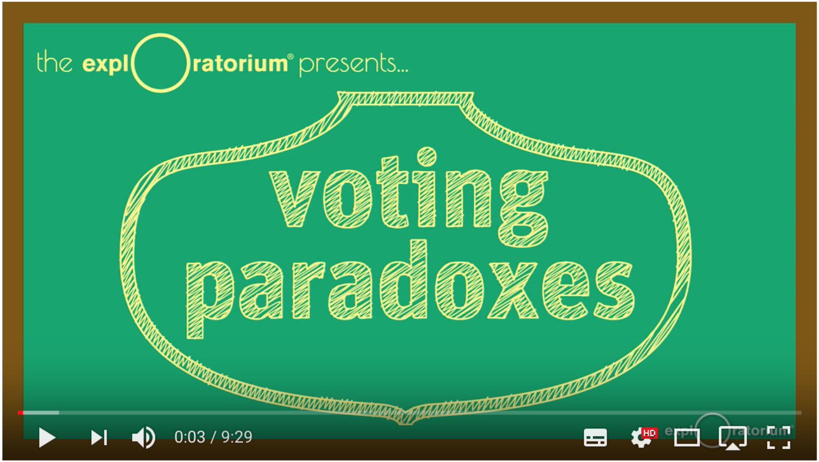
Course Overview

- Voting
- Judgment Aggregation
- Stable Matching

- Throughout everything, we will discuss complexity theory

Voting

Voting



Social Choice Functions and Social Welfare Functions

- a set $N = \{1, \ldots, n\}$ of voters
- a set $A = \{a_1, \ldots, a_m\}$ of alternatives (or candidates)
- $\mathcal{L}(A)$ denotes all linear orders $\succ$ over A
  - (linear order: transitive, antisymmetric, complete relation)
- a profile $P \in \mathcal{L}(A)^n$ consists of a linear order for each voter

- a social welfare function $f : \mathcal{L}(A)^n \to \mathcal{L}(A)$ takes a profile and outputs a social preference order
- a social choice function $f : \mathcal{L}(A)^n \to 2^A \setminus \emptyset$ takes a profile and outputs a nonempty set of winners

Example

- $N = \{1, 2, 3, 4, 5, 6\}$
- $A = \{c$ 'chocolate', s 'strawberry', v 'vanilla'$\}$
- Profile P:

P	1	2	3	4	5	6
#1	c	c	c	v	v	s
#2	s	s	s	c	c	c
#3	v	v	v	s	s	v

- I.e., $P = (\succ_1, \succ_2, \succ_3, \succ_4, \succ_5, \succ_6)$, where:

$$\begin{aligned} \succ_1 = \succ_2 = \succ_3 = &\quad \{(c, s), (c, v), (s, v)\} \\ \succ_4 = \succ_5 = &\quad \{(v, c), (v, s), (c, s)\} \\ \succ_6 = &\quad \{(s, c), (s, v), (c, v)\} \end{aligned}$$

The Plurality Rule

- The Plurality rule is the social choice function that selects all candidates that are ranked $\#1$ the most times

  (with highest plurality score)

- Example:

P	1	2	3	4	5	6
$\#1$	c	c	c	v	v	s
$\#2$	s	s	s	c	c	c
$\#3$	v	v	v	s	s	v

$$\text{Plurality}(P) = \{c\}$$

Instant-Runoff Voting (IRV)

- IRV is the social choice function that selects a winner as follows:
  - Repeat:
    - Count the plurality score of each alternative
    - If some alternative is ranked #1 by a majority of voters, this is the winner
    - Otherwise, remove the voter with lowest plurality score from the profile (use a tie-breaking if there are more)

- Example:

P	1	2	3	4	5	6
#1	c	c	c	v	v	s
#2	s	s	s	c	c	c
#3	v	v	v	s	s	v

$$\text{IRV}(P) = \{c\}$$

Condorcet Extensions

- A Condorcet winner for a profile P is an alternative $a \in A$ such that for each alternative $b \in A$ with $a \neq b$, a strict majority of voters prefers a to b
- Not for every profile a Condorcet winner exists
- Example:

P	1	2	3
#1	c	v	s
#2	s	c	v
#3	v	s	c

- A social choice function that selects the Condorcet winner as unique winner, if it exists, is called a Condorcet extension

The Kemeny Rule

- The Kemeny rule (or Kemeny-Young rule) is a Condorcet extension
- The Kendall-Tau distance $d(\succ_1, \succ_2)$ between two rankings $\succ_1, \succ_2 \in \mathcal{L}(A)$ is the number of pairs $(a, b) \in A \times A$ on which $\succ_1$ and $\succ_2$ disagree
- Consider all $\succ \in \mathcal{L}(A)$ that minimize:

$$\sum_{i \in N} d(\succ, \succ_i).$$

- The Kemeny rule selects the top candidate from each $\succ$ minimizing the total Kendall-Tau distance to P as a winner

The Kemeny Rule

- Example:

P_1	1	2	3
#1	c	v	s
#2	s	c	v
#3	v	s	c

$$\text{Kemeny}(P_1) = \{c, s, v\}$$

- Example:

P_2	1	2	3	4	5	6
#1	c	c	c	v	v	s
#2	s	s	s	c	c	c
#3	v	v	v	s	s	v

$$\text{Kemeny}(P_2) = \{c\}$$

The Borda Rule

- The Borda rule is a social choice function that is based on the Borda score:
- Net preference of a over b:

$$\mathrm{Net}_P(a \succ b) = |\{\, j \in N : a \succ_j b \,\}| - |\{\, j \in N : b \succ_j a \,\}|.$$

- Borda score of a:

$$\mathrm{Borda}_P(a) = \sum_{\substack{b \in A \\ a \neq b}} \mathrm{Net}_P(a \succ b).$$

- The Borda rule selects the candidates with highest Borda score as winners

The Borda Rule

- Example:

P_1	1	2	3
#1	c	v	s
#2	s	c	v
#3	v	s	c

$$\text{Borda}(P_1) = \{c, s, v\}$$

- Example:

P_2	1	2	3	4	5	6
#1	c	c	c	v	v	s
#2	s	s	s	c	c	c
#3	v	v	v	s	s	v

$$\text{Borda}_P(c) = 9, \quad \text{Borda}_P(s) = 5, \quad \text{Borda}_P(v) = 4$$

$$\text{Borda}(P_2) = \{c\}$$

The Axiomatic Approach

- Are some voting rules better than others?
- This question has been investigated with the axiomatic approach: mathematically specify normatively appealing axioms, and find out which voting rules satisfy these
- Examples of axioms (for SWFs):
  - Anonymity: "changing the order of voters in the profile doesn't change the outcome"
  - Weak Pareto efficiency: "if all voters in the profile prefer a to b, then a is preferred to b in the social preference order"
  - Independence of Irrelevant Alternatives (IIA): "the relative ranking of a and b in the social preference order depends only on the relative ranking of a and b in all individual preferences in the profile"
  - etc.

Arrow's Theorem

- Seminal result in social choice theory: Arrow's Theorem

Theorem (Arrow, 1951)

When there are three or more alternatives, then every social welfare function that satisfies weak Pareto efficiency and IIA must be a dictatorship.

- A SWF is a dictatorship if there is one voter whose preference order it always outputs as social preference order

- Takeaway message: there is no best voting rule, that satisfies all desirable properties simultaneously

Problems Studied in Voting

- Besides normative axioms, computational properties of voting rules are relevant factors for choosing between them
- Several computational tasks are relevant:
  - Winner determination: given a profile P, determine the winner(s)
  - Strategic manipulation: given a profile P, can voter i report a false preference order to get a more preferred outcome?
  - Bribery: given a profile P, can one change the preference order of at most m individuals to make a certain candidate a the winner?
  - etc.

Complexity Theory

What is Computational Complexity?

- The study of what you can compute with limited resources
  - Resources, e.g.: time, memory space, random bits
- Includes determining the practical limits on what you can do with computers
- Distinguish different degrees of computational difficulty
- Central question: the P versus NP problem
  (one of the $1 Million *Millennium Prize Problems*)

How to Measure Complexity

- Computational problems are modelled as input-output mappings
- Inputs are strings (over a finite alphabet)
  - Such a string can encode all kinds of objects, e.g., a graph:

    ```
    node(1).  node(2).  node(3).
    edge(1,2).  edge(2,3).
    ```
  - We often switch perspectives between strings and objects
- Measure the complexity (e.g., running time) of an algorithm by the number of computation steps taken as a function of the input size n
  - E.g., on inputs of size n, the algorithm takes $f(n) = 2 \cdot n^2$ steps

How to Measure Complexity

- We typically use a worst-case perspective (for algorithms):
  - The function $f(n)$ measuring the (time) complexity of an algorithm expresses the maximum complexity over all inputs of size n
- We say that a function $f(n)$ expresses the complexity of a problem, if there exists some algorithm that solves the problem and that is of complexity $f(n)$
- Example:
  - A problem Q is solvable in time n^2
    if there exists an algorithm solving the problem
    such that for all inputs x it takes (at most) $|x|^2$ time steps

Polynomial-time vs. Exponential-time

- There is an important difference between algorithms that run in time, say, n^2 vs. algorithms that run in time, say, 2^n
- Illustration (time needed for 10^{10} steps per second):

n	n^2 steps	2^n steps
2	0.00000002 msec	0.00000002 msec
5	0.00000015 msec	0.00000019 msec
10	0.00001 msec	0.0001 msec
20	0.00004 msec	0.10 msec
50	0.00025 msec	31.3 hours
100	0.001 msec	9.4×10^{11} years
1000	0.100 msec	7.9×10^{282} years

Big O Notation

- In order to abstract away from constants

  (that are often immaterial in the difference between polynomial time vs. exponential time)

  often Big O notation is used:

  - Let $f, g : \mathbb{N} \to \mathbb{N}$ be functions
  - We say that $f(n)$ is $O(g(n))$ if there exists some $n_0 \in \mathbb{N}$ and some constant c such that for all $n \geq n_0$ it holds that $f(n) \leq c \cdot g(n)$
  - Example: $2n^2 + 3n$ is $O(n^2)$

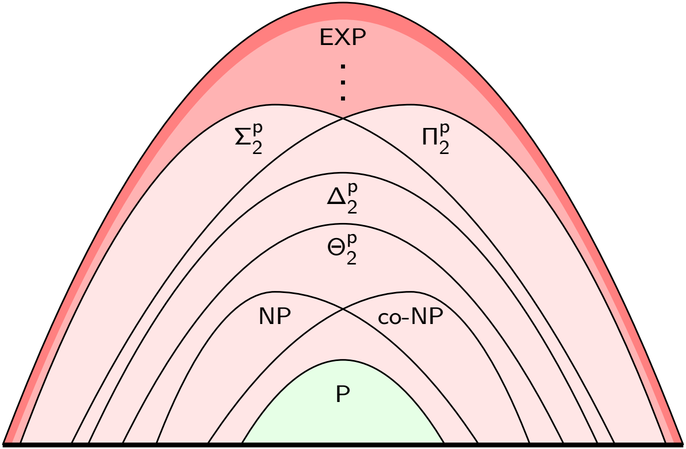
Decision Problems

- To make the analysis easier, we often restrict attention to decision problems:
  - Decision problems are input-output problems where the output is always 0 or 1 ("no" or "yes")
- Alternatively, one can see decision problems as formal languages
  - Let Σ be the (finite) alphabet
  - Then Σ^* is the set of all finite strings over Σ all possible inputs
  - A decision problem $Q \subseteq \Sigma^*$ is a formal language consisting of all inputs for which the answer is 1 (or "yes")

Complexity Classes

- A complexity class is a set of decision problems (that are of related complexity)
- The class P is the set consisting of all decision problems $Q \subseteq \Sigma^*$ that are solvable in polynomial time, i.e., in time $O(n^c)$, for some constant $c \in \mathbb{N}$
- The class NP is the set of all decision problems $Q \subseteq \Sigma^*$ for which there exists a polynomial function $q : \mathbb{N} \to \mathbb{N}$ and a polynomial-time algorithm V, such that for all inputs $x \in \Sigma^*$:
  - if $x \in Q$, then there is some string $y \in \Sigma^{q(|x|)}$ such that V outputs 1 on input (x, y), and
  - if $x \notin Q$, then for all strings $y \in \Sigma^{q(|x|)}$ it holds that V outputs 0 on input (x, y).

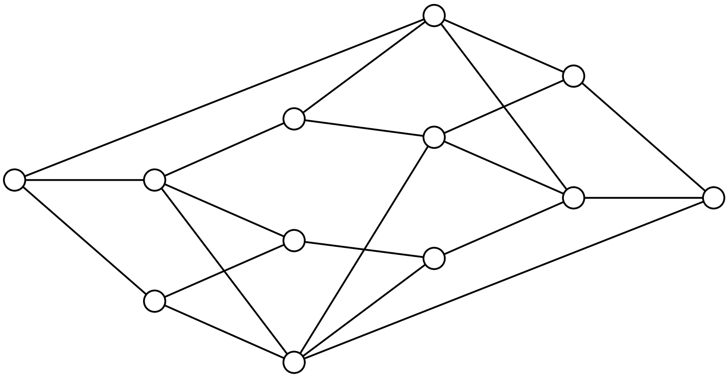
Different levels of hardness



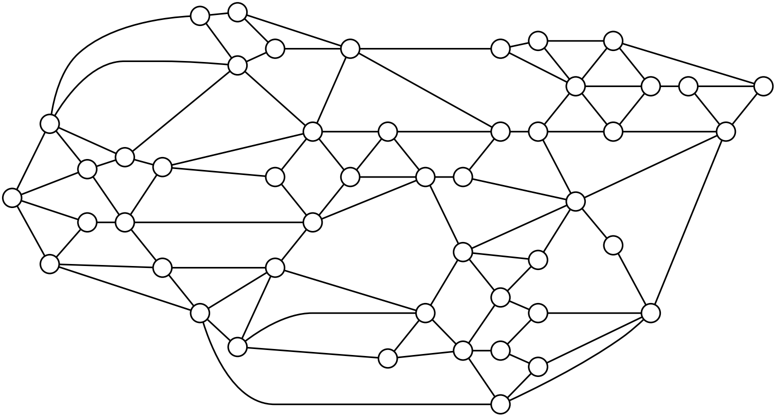
Example: Graph c-Coloring

- 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 $\{1, 2, \ldots, c\}$ so that no two connected nodes have the same color

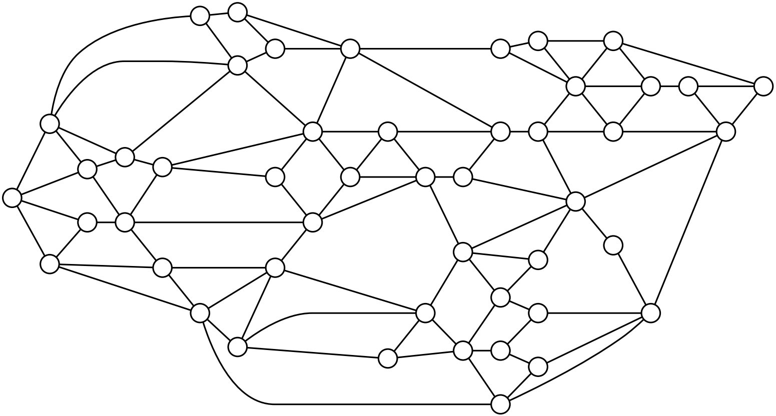
Color this graph with 2 colors!



Color this graph with 2 colors!



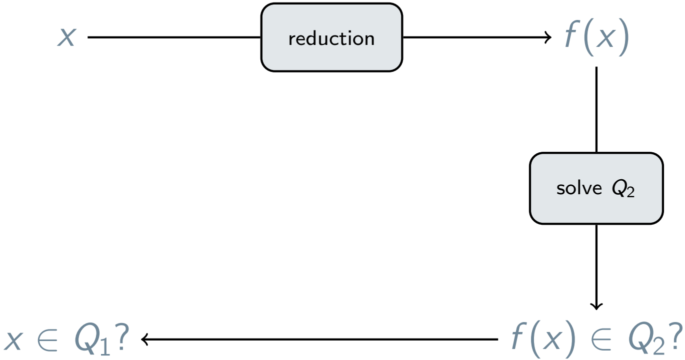
Now, color this graph with 3 colors!



NP-hardness and -completeness

- To give evidence that a problem is not polynomial-time solvable, we use the notion of hardness
- A (polynomial-time) reduction from one problem $Q_1 \subseteq \Sigma^*$ to another problem $Q_2 \subseteq \Sigma^*$ is a function $f : \Sigma^* \to \Sigma^*$ such that:
  - for each $x \in \Sigma^*$, it holds that $x \in Q_1$ if and only if $f(x) \in Q_2$, and
  - $f(x)$ is computable in time $O(|x|^c)$, for some constant c
- A problem $Q \subseteq \Sigma^*$ is NP-hard if for all problems $Q' \in \text{NP}$ there is a reduction from Q' to Q
  - If you can solve Q in polynomial time, then you can solve all problems in NP in polynomial time!
- A problem is NP-complete if it is both NP-hard and in NP

Reductions



The Cook-Levin Theorem

Theorem (Cook, 1971; Levin, 1973)

There are NP-complete problems.
In particular, SAT is NP-complete.

SAT:

- *Input: a propositional logic formula φ.*
- *Output: is φ satisfiable?*

- It is widely believed (but not proven) that $\mathrm{P} \neq \mathrm{NP}$
- Under the assumption that $\mathrm{P} \neq \mathrm{NP}$:
  if a problem is NP-hard, it is not in P

Example of a Reduction

- Reduction from 3-Coloring to SAT:
  - Let $G = (V, E)$ be a graph, with $V = \{v_1, \ldots, v_n\}$.
  - Construct $f(G) = \varphi$ to be the conjunction of the following formulas.
  - $\mathrm{Var}(\varphi) = \{\, x_{i,c} : 1 \leq i \leq n, c \in \{\mathrm{r}, \mathrm{g}, \mathrm{b}\}\,\}$
  - For each $v_i \in V$, add:

    $$(x_{i,\mathrm{r}} \vee x_{i,\mathrm{g}} \vee x_{i,\mathrm{b}}), (\neg x_{i,\mathrm{r}} \vee \neg x_{i,\mathrm{g}}), (\neg x_{i,\mathrm{r}} \vee \neg x_{i,\mathrm{b}}), (\neg x_{i,\mathrm{g}} \vee \neg x_{i,\mathrm{b}})$$

  - For each $\{v_i, v_j\} \in E$, add:

    $$(\neg x_{i,\mathrm{r}} \vee \neg x_{j,\mathrm{r}}), (\neg x_{i,\mathrm{g}} \vee \neg x_{j,\mathrm{g}}), (\neg x_{i,\mathrm{b}} \vee \neg x_{j,\mathrm{b}})$$

Voting & Complexity Theory

Complexity of Winner Determination

- What is the complexity of the winner determination problem for the different voting rules that we saw before?

- Let F be a voting rule:

WinDet(F)

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

Output: Is $a^* \in F(P)$?

Example: Encoding of a Profile as a String

P	1	2	3	4	5	6
#1	c	c	c	v	v	s
#2	s	s	s	c	c	c
#3	v	v	v	s	s	v

```
voters(1,2,3,4,5,6).
candidate(c,s,v).
pref(1,c,s,v).
pref(2,c,s,v).
pref(3,c,s,v).
pref(4,v,c,s).
pref(5,v,c,s).
pref(6,s,c,v).
```

Complexity of Winner Determination (Plurality)

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

- Which candidates are the winners for this profile P for the Plurality rule?

$$b$$

Complexity of Winner Determination (Plurality)

Proposition

Winner Determination for the Plurality voting rule is polynomial-time solvable.

Complexity of Winner Determination (IRV)

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

- Which candidates are the winners for this profile P for IRV?

$$a$$

Complexity of Winner Determination (IRV)

Proposition

Winner Determination for the IRV voting rule is polynomial-time solvable.

Complexity of Winner Determination (Borda)

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

- Which candidates are the winners for this profile P for the Borda rule?

a

Complexity of Winner Determination (Borda)

Proposition

Winner Determination for the Borda voting rule is polynomial-time solvable.

Complexity of Winner Determination (Kemeny)

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

- Which candidates are the winners for this profile P for the Kemeny rule?

$$a$$

Complexity of Winner Determination (Kemeny)

Theorem (Hemaspaandra, Spakowski, Vogel, 2005)

Winner Determination for the Kemeny voting rule is Θ_2^p-complete.

E. Hemaspaandra, H. Spakowski, and J. Vogel. The Complexity of Kemeny Elections. Theoretical Computer Science, 349(3), 382–391, 2005.

Complexity as a Criterion

- Computational complexity considerations also play a role in choosing which voting rule to use for your application
  - Winner determination problem
  - Other problems (more on this tomorrow)
  - More complexity tools (more on this tomorrow)

Recap

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

Homework exercise

- Find a polynomial-time reduction from 3SAT to 3-Coloring.
- (Hint: look at Section 4.3 of the following book.)

O. Goldreich. P, NP, and NP-Completeness: the Basics of Computational Complexity. Cambridge University Press, 2010.