Seminar on Computational Social Choice

Autumn 2016

Matti Järvisalo and Johannes Wallner

Practical arrangements, introduction, choosing topics

September 8, 2016

Practical Arrangements

Course Information

Instructors:	Matti Järvisalo, Johannes Wallner

{matti.jarvisalo,johannes.wallner}@cs.helsinki.fi

Credit units:	3 ECTS
Language:	english
WWW:	https://www.cs.helsinki.fi/en/courses/58316302/2016/s/s/1
Announcements:	seminar webpage, email
Reception:	Contact instructor(s) by email for an appointment. or during seminar meetings

Course Requirements

- Choose a topic (scientific chapter/article) to study
- Write a 10-15 page (plus references) report on the topic
- Give a 40-min presentation on the topic
- Give constructive feedback on another student's report (and draft)
- Act as the opponent of another student's presentation
- Actively attend the seminar

Grading:

- On scale 0–5
- Report 50%, presentation 50%
- Activity (incl. being an opponent) ± 1 grade

Deadlines

- All deadlines are strict — you will fail the course if you do not meet a deadline
  - Need proof of illness to postpone deadlines
- September 15: choose topic and presentation date
- Presentations:
  Two presentations per week during period II, Nov 3 – Dec 15
- Friday before your presentation: preliminary report & slides
  (send to instructors and your opponent by email)
- At presentation day:
  Presenters: arrive early to set-up slides
  opponents:
  - Actively ask questions during/after presentation
  - Give constructive feedback on report draft,
    send to presenter and instructors by email
- December 18: final report

Choosing a Topic

Handbook of Computational Social Choice

- Book available through
  `http://www.cambridge.org/download_file/898428`
  - Password for the pdf: cam1CSC
- Each chapter 2–19 one possible topic
- Can suggest a topic outside the list!
- You may need to read additional articles for necessary background
- Reserve topic no later than Wed 15
  - Preferably already today!

Report

- A seminar report is a short review paper:
  you explain some interesting results in your own words.
- A typical seminar report will consist of the following parts:
  - an informal introduction,
  - a formally precise definition of the problem that is studied,
  - a brief overview of very closely related work— here you might cite approx. 3–10 papers and explain their main contributions,
  - a more detailed explanation of one or two interesting results, with examples
  - conclusions.
- Superficially, your report should look like a typical scientific article.
  - However, it will not contain any new scientific results, just a survey of previously published work.

Presentation

- The presentation is an overview of the report
    - You should understand what you are saying
    - Everyone should understand you
    - The abstraction level should be right
    - Examples are always good to communicate ideas

Templates

- Use of Latex especially for the seminar report is strongly encouraged
- Latex template for the report available via the seminar webpage
- For the presentation, use software of your choice
  - If you use latex, look into the beamer package

Some Words of Advice

- Start working on your topic early!
- Depending on your background, you will very likely need to read additional papers for background
- Aim at understanding the key aspects of your topic – do not get side-tracked
- You are responsible for figuring out the details
  - The instructors will not teach you all necessary background
  - In case you get completely stuck, contact the instructors
  - You will need to show that you have made a serious attempt to understand the topic by yourself

Introduction

Social Choice Theory

Social choice theory studies

"aggregation of individual preferences towards a collective choice"
Brandt et al. *Handbook of Computational Social Choice*

Related disciplines	**Computational aspects**
Political science	computational complexity
Economics	exact algorithms
Logic	approximability
Mathematics	AI
Operations research	Multi-agent systems
Computer science	...

Potential applications of computational social choice in

- decision-making technologies
- policy making
- distributed computing

Pliny the Younger: Decide Fate of Prisoners

- Options:
  - (A) Acquittal
  - (B) Banishment
  - (C) Condemnation to death
- (A) had majority
- Proponent of a harsh punishment removed option (C)
- Lead to rallying behind (B)
- Nowadays known as election control by deleting candidates
- Potential for strategic manipulation

Borda's Rule and Condorcet

- Jean-Charles de Borda (1733–1799) proposed to "Borda rule"
- Example: 11 voters, 3 candidates

4	3	2	2
Peter	Paul	Paul	James
Paul	James	Peter	Peter
James	Peter	James	Paul

- Points for each candidate ranked less
- Winner: Paul ($4 \cdot 1 + 3 \cdot 2 + 2 \cdot 2 + 2 \cdot 0 = 14$)
- Marquis de Condorcet's observation:
- Paul wins, but 6 out 11 voters prefer Peter over Paul

Pairwise Majority / Condorcet Winner

- What about pairwise majority?

4	3	2	4
Peter	Paul	Paul	James
Paul	James	Peter	Peter
James	Peter	James	Paul

- Majority prefer
  - Peter to Paul (8)
  - Paul to James (9)
  - James to Peter (7)
- $\Rightarrow$ resulting preference relation cyclic

Arrow's Theorem

- Kenneth Arrow, 1951: Mathematical framework for social choice
- Ingredients
  - $N = \{1, \ldots, n\}$ voters
  - set A of candidates
  - preference relation (linear order) $\succsim_i$ from each voter
- Aggregated result: social welfare function (SWF) $\succsim$
- Formal properties
  - weakly Paretian
  - independent of irrelevant alternatives (IIA)
  - dictatorship

Every SWF with 3 or more candidates that satisfies weakly Paretian and IIA is a dictatorship.

Computational Social Choice

From axiomatic/normative treatment to computational study of social choice

- Practical applicability of e.g. voting rules
  - Requirements / formal properties
  - Implementability
- Algorithms — exact/approximate
- Computational complexity
  - "Cost" of computing result
  - "Barriers" for manipulation
- ...

Some key topics:
Voting, fair allocation, coalition formation, judgment aggregation, group recommendation, crowdsourcing, ...

Choosing Topics & Dates

Voting — Fair Allocation — Coalition Formation — Additional Topics

Topic List

Part 1 Voting

2 **Introduction to the theory of voting**
3 **Tournament solutions**
4 **Weighted tournament solutions**
5 Dodgson's rule and Young's rule — topic reserved / Berg
6 Barriers to manipulation in voting — topic reserved / Leinonen
7 **Control and bribery in voting**
8 Rationalizations of voting rules — topic reserved / Sallinen
9 **Voting in combinatorial domains**
10 **Incomplete information and communication in voting**

Topic List (contd.)

Part 2 Fair Allocation

11	Introduction to the theory of fair allocation	topic reserved / Viinikka
12	Fair allocation of indivisible goods	topic reserved / Mertanen
13	Cake cutting algorithms	topic reserved / Karikoski

Part 3 Coalition Formation

14	Matching under preferences	topic reserved / Zosa
15	Hedonic games	topic reserved / Linkola
16	Weighted voting games	topic reserved / Lin

Topic List (contd.)

Part 4 Additional Topics

17 **Judgment aggregation**
18 **The axiomatic approach and the internet**
19 **Knockout tournaments**

Other Outside the handbook

- Felix Brandt, Christian Geist: *Finding Strategyproof Social Choice Functions via SAT Solving*. Journal of Artificial Intelligence Research 55:565-602 (2016) topic reserved / Niskanen

Refresher on Computational Complexity

Decision Problems

A computation problem consists of:

- an instance of the problem (the input instance)
- a question applicable to any instance of the problem

- For decision problems, the question has a yes/no answer for any instance of the problem.
- An algorithm that can provide the correct answer to any instance of a decision problem B is called a decision procedure for B.
  - Such an algorithm is said to decide B.
- Fundamentally, only problems with an infinite number of instances are interesting (Why?)

Decision Problems: Examples

- k-COLORING:
  INSTANCE: A graph $G = (V, E)$ and a positive integer k.
  QUESTION: Is G k-colorable?
- SAT:
  INSTANCE: A propositional formula F in conjunctive normal form (CNF).
  QUESTION: Is F satisfiable?

Other Types of Computational Problems

In search, optimization, counting, and enumeration problems, the question and answer to an instance I are more complicated.

Search problems

Find a solution to I (a witness for the "yes" answer of the decision problem; answering "no" if there is no solution).
Also known as function problems.

Optimization problems:

Find a best solution to I,
minimizing or maximizing some cost function over all solutions.

Counting problems:

Count the number of solutions to I.

Enumeration problems:

List all solutions to I.

Some Problems are Easy, Some are Hard

Problem P is computationally easy:

There is a deterministic polynomial-time algorithm for P (i.e., an algorithm whose running time is polynomially-bounded w.r.t. the input instance size.

Problem P is (seems) computationally hard:

No deterministic polynomial-time algorithm is known.

$\Rightarrow$ *For any known algorithm A, there is an infinite number of instances of increasing size on which the running time of A increases super-polynomially w.r.t. the input instance size.*

P v NP

- Computational complexity theory:
  Categorization of problems into problem classes
- The most famous and often practically most relevant distinction is between the problem classes **P** and **NP**
  - **P** contains all decision problems for which there are deterministic polynomial-time algorithms.
  - **NP** contains decision problems for which there are small (polynomial-size) certificates, i.e., any possible solution candidate can be checked in polynomial time.

P v NP

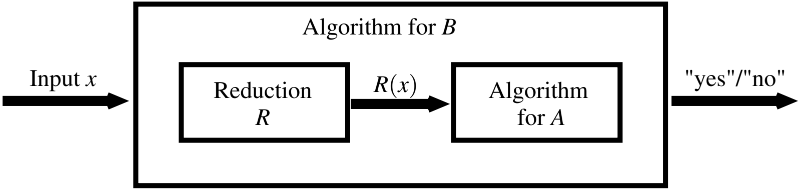
- The "**P** = **NP**?" question is still unresolved.
  - If the verification of a solution is easy,
    finding a solution may still not be easy.
- **NP** contains a vast number of hard decision problems that have a lot of practical relevance.

NP-Completeness

- NP-complete problems are <u>the hardest</u> within NP
- <u>Completeness = hardness + inclusion</u>
- To prove a problem NP-complete:
  - Reduce some NP-complete problem to your problem (NP-hardness)
  - Argue that a given solution candidate can be guessed and checked in polynomial time (inclusion in NP)
- NP-complete problems: SAT, graph coloring, . . .

Relating Problems: Reductions

- The relationship between two decision problems A and B can be studied via reductions
- B reduces to A:
  There is a transformation (reduction) R which,
  given any instance x of B,
  produces an input instance $R(x)$ of A for which the following holds:
  $R(x)$ is a "yes"-instance of A if and only if x is a "yes"-instance of B.



- Typically reductions are required to be computable in polynomial time — a necessity for NP-hardness proofs

P, NP, and Beyond

- There are infinitely many problems which may be harder than **NP** — polynomial hierarchy
- L $\subseteq$ P $\subseteq$ NP $\subseteq$ NP^{NP} $\subseteq \cdots \subseteq$ PSPACE