Going Beyond the Traditional Complexity Analysis

Ronald de Haan

Technische Universität Wien

ALGORITHMS AND
COMPLEXITY GROUP

What will I talk about?

- Investigating computational complexity issues in computational social choice using tools capable of subtler analysis leads to more useful insights

- Classical complexity theory has a negative bias

- This is problematic when using intractability as a barrier

- What to watch out for when using finer tools to reduce this negative bias

What's Next?

Complexity in Computational Social Choice

- Computational complexity is an important aspect of research in computational social choice
  - To know in what cases some algorithmic approaches are possible
    - e.g., efficient algorithms for computing the winner of an election
  - To know in what cases some algorithmic approaches are impossible
    - e.g., using intractability results as a barrier against strategic behavior

General Methodology of Computational Complexity

- Theoretical, mathematical framework to classify how hard it is to solve computational problems
  - distinguish tractable from intractable

- To do this productively, an abstract model is used:
  - Inputs are strings over some alphabet
  - Consider how the running time grows with the input size n
  - For each n, count the maximum over any input of size n

- Theoretical distinction between P and NP-hard (or worse)
  - Idea: this distinction corresponds by and large to the border between tractable and intractable in practice

What's Next?

Intractability Results can be Overly Negative

- Intractability results (e.g., NP-hardness) indicate that all algorithms require exponential time in the worst case
  - i.e., there is no algorithm that is efficient for all inputs
- There could still be an algorithm that works efficiently for a large subclass of inputs
- In fact, for many NP-complete problems, important classes of inputs have been found with efficient algorithms
  - Vertex Cover: given a graph $G = (V, E)$, is there a subset of vertices of size m that touches each edge?
  - If m is small, this can be solved, even for large graphs

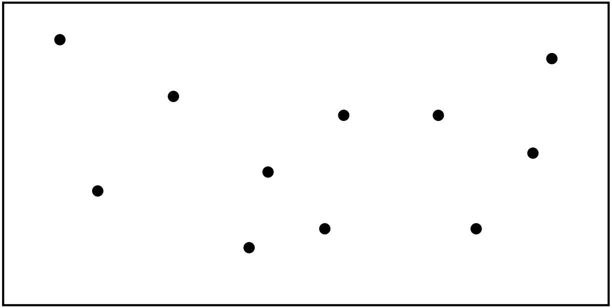
Example: Manipulation in Voting

- Consider voting using Single Transferable Vote (STV)
- The problem of manipulation is NP-complete
  - So: each algorithm to compute a manipulation policy takes time $2^{\Omega(n)}$
- However, this could give a **false sense of safety!**
- Strategic manipulation is solvable in time $m! \cdot \mathrm{poly}(n)$, where m is the number of candidates
  - Whenever m is small, this is feasible

The Need for Stronger Intractability Results

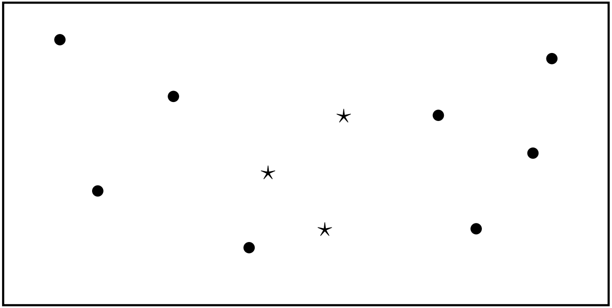
- Negative bias in intractability results:
  - Good when looking for algorithms that are (guaranteed to be) efficient
- When using intractability to argue that strategic behavior is obstructed, it is naive to disregard this bias
- Classical theory of computational complexity gives weak intractability results
- We need **stronger** intractability results to argue for complexity barriers against strategic behavior

The Need for Stronger Intractability Results.. in pictures



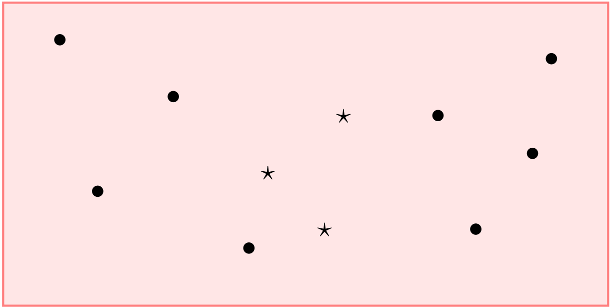
Imagine these are all possible inputs

The Need for Stronger Intractability Results.. in pictures



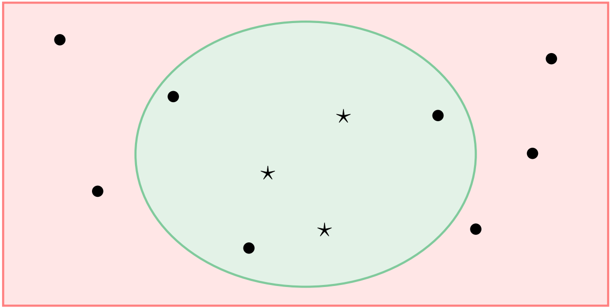
Suppose you care about the $\star$'s

The Need for Stronger Intractability Results.. in pictures



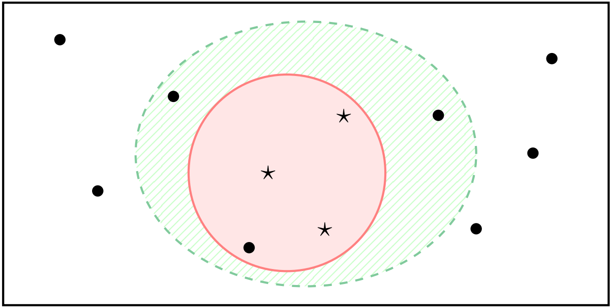
Classical intractability..

The Need for Stronger Intractability Results.. in pictures



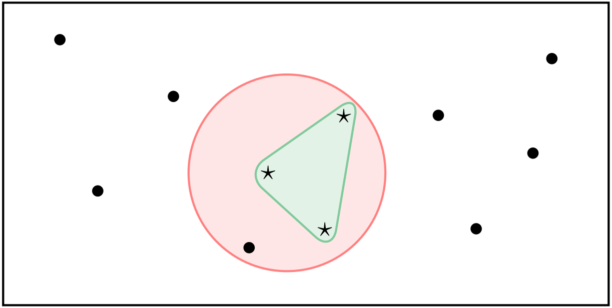
does not rule out an efficient algorithm for a subset of inputs

The Need for Stronger Intractability Results.. in pictures



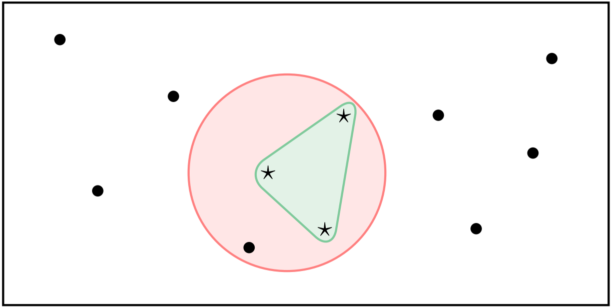
Stronger intractability results can rule out such an algorithm

The Need for Stronger Intractability Results.. in pictures



But careful.. there could always be algorithms
for more restricted classes of inputs

The Need for Stronger Intractability Results.. in pictures



Intuitively, stronger intractability results leave less space for undiscovered efficient algorithms

A typical pitfall

- When modelling, you consider an abstraction of the scenario you care about
- Typically, you 'err' on the side of generality
- Example: judgment aggregation, with issues $\varphi_1, \ldots, \varphi_n$ represented by **arbitrary** propositional formulas
- In this setting, intractability results are **everywhere!**
- These results might not say much if your scenario contains logical relations expressible by statements of the form: (if a, then b)

What's Next?

Going Beyond Showing NP-Hardness

- Investigate the possibilities of more intricate methods offered by (theoretical) computer science, e.g.:
  - Investigate the problem for fragments of inputs
  - Consider approximation algorithms
  - Use the framework of parameterized complexity theory
  - Encode problem inputs into SAT, and use SAT solvers
  - Employ typical-case complexity
  - Empirically investigate how algorithmic methods perform

Parameterized Complexity in a Nutshell

- Traditional complexity theory measures running time only in terms of the input size n
- To reduce the worst-case negative bias, take into account more than just this number n:
- Parameterized complexity measures running times in terms of input size n and a parameter k
- The parameter captures *structure* that is present in the input
  - the smaller k, the more structure
- Fixed-parameter tractability: running time of $f(k) \cdot \mathrm{poly}(n)$, for some (computable) function f
- (Idea: worst-case over inputs of size n and with small k)

People have been parameterizing...

- Many parameterized complexity results in computational social choice
  - parameters, e.g.: number of candidates, number of voters

- Example: manipulation of STV with a small number of candidates
  - Relativizes the NP-hardness result for manipulation of STV
  - Solvable in time $m! \cdot \mathrm{poly}(n)$, where m is the number of candidates
  - Whenever m is small, this is feasible

But stay on your toes!

- Remember from the picture:
  - Whenever you have intractability for a 'small circle'..
  - there could be an efficient algorithm for an 'even smaller circle'
- Parameterized intractability results still have a negative bias
- Keep trying to 'reduce the circle' as much as possible for your application:
  - Add another parameter
  - Restrict to a fragment of the inputs
  - ...

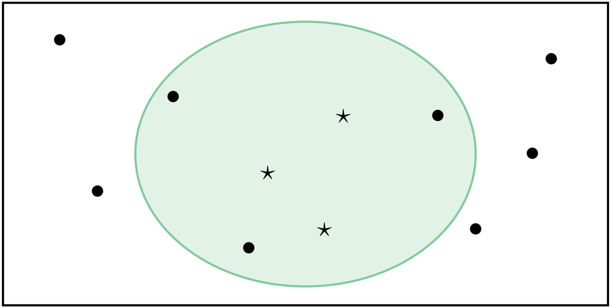
Example: Manipulating the Kemeny JA Procedure

- Strategic manipulation: can an individual report an insincere judgment to obtain a *better* group outcome?
- For the Kemeny JA procedure, this is Σ_2^p-complete
  - (worse than NP-complete..)
- Even when parameterized by # of issues, the problem remains intractable
- **But**, with if-then logical constraints, manipulation becomes tractable

Stricter Parameters

- For tractability results, loose (or general) parameters—as few as possible together—are ideal

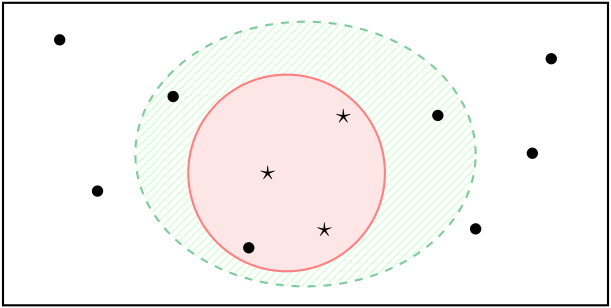
Stricter Parameters



Stricter Parameters

- For tractability results, loose (or general) parameters—as few as possible together—are ideal
- For intractability results, combinations of as many as possible strict parameters are ideal

Stricter Parameters



Stricter Parameters

- For tractability results, loose (or general) parameters—as few as possible together—are ideal
- For intractability results, combinations of as many as possible strict parameters are ideal
- To get useful, strong intractability results, consider parameters that severely restrict structure, e.g.:
  - treewidth (and others, e.g., clique-width)
  - backdoors
  - distance to single-peakedness or unidimensional alignment
  - maximum distance between any two votes
  - maximum range of candidates (in voting)

Strict Parameters for the Example of Kemeny in JA

- Example of judgment aggregation
  - restriction: all issues are propositional variables
  - restriction: integrity constraint contains no additional variables
  - parameter: maximum distance h between any two individual judgments
  - parameter: number p of individuals
- Intractability in this constrained setting is much more powerful than
- **But**: even in this case, further restrictions could lead to efficient algorithms

Combining Various Methods

- Negative theoretical results rule out one type of algorithmic approach
- Idea of strong negative results: try to rule out approaches that are as specialized as possible
- Investigate combinations of algorithmic methods
- For example: combination of parameterized algorithms and SAT encodings

SAT Encodings

- General idea:
  - encode your problem input as an instance of SAT
  - use a SAT solving algorithm to solve the problem
- Applicable for problems in NP
- No worst-case guarantees
- Works well in many practical settings

Possibilities for the Example of Kemeny in JA

- Previous example: strategic manipulation in judgment aggregation for the Kemeny procedure
  - parameter: number of issues
- Parameterized intractability result: no efficient parameterized algorithm for this case
- **But**: can be solved in fixed-parameter tractable time by encoding into a small number of SAT instances

The Holy Grail

- Study your application setting
- Consider **all** algorithmic methods (that we know about)
- Give theoretical (intractability) results that these methods are not possible in your application
- Give experimental evidence that these methods really do not work well

- (Compare this to just showing NP-hardness)

What's Next?

Overview

Complexity in Computational Social Choice

Negative Bias of Complexity Theory

What to do?

Summary

Summary

- Investigating computational complexity issues in computational social choice using tools capable of subtler analysis leads to more useful insights

- Classical complexity theory has a negative bias

- This is problematic when using intractability as a barrier

- Use finer tools to reduce this negative bias

- Be much more demanding when using negative results as a barrier!!

Thanks!

Questions?

Table of Contents