1 Motivation & Outline

22 April 2025

Prof. Dr. Sebastian Wild

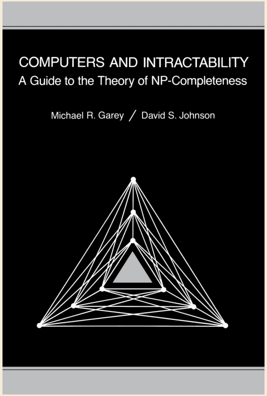
CS627 (Summer 2025)
Philipps-Universität Marburg

version `2025-04-22 09:47`

Outline

1.1 How to Solve Hard Problems?

View on NP-completeness in introductory courses



"I can't find an efficient algorithm, but neither can all these famous people."

Garey, Johnson 1979

. . . but this is not the end of the story!

*"you had just neatly sidestepped potential charges of incompetence by proving that the bandersnatch problem is NP-complete. However, the bandersnatch problem had **refused to vanish at the sound of those mighty words**, and you were still faced with the task of finding some usable algorithm for dealing with it."*

Garey, Johnson 1979, Chapter 6

. . . but this is not the end of the story!

"you had just neatly sidestepped potential charges of incompetence by proving that the bandersnatch problem is NP-complete. However, the bandersnatch problem had ***refused to vanish at the sound of those mighty words****, and you were still faced with the task of finding some usable algorithm for dealing with it."*

Garey, Johnson 1979, Chapter 6

$\rightsquigarrow$ Look for "loopholes" in the impossibility results:

- hard only for large integers in input? $\rightsquigarrow$ *pseudopolynomial* algorithms
- hard only for contrived instances? $\rightsquigarrow$ *fixed-parameter* algorithms
- hard only for exact solution? $\rightsquigarrow$ *approximation* algorithms
- hard only in rare cases? $\rightsquigarrow$ *randomized* algorithms
- hard only without a quantum computer? $\rightsquigarrow$ *quantum* algorithms (not here)
- . . .

also quantum computers are not here

Algorithms for hard problems

- for each possible loophole left by NP-hardness, we consider
  - algorithmic techniques that exploit the loophole
  - impossibility results showing when that exploit fails to apply
  - examples of problems where this loophole does (not) help

- focus is on provable results & eternal truths
  theoretical algorithms and complexity theory
  - strong foundation in math and theoretical computer science needed
  - ultimate goal is a practically relevant solution,
    but will often strive for simpler, "pedagogical" examples here

Loophole 1: Large integers

- **Loophole:** problem only hard when the integers that are part of the input are huge
- **Algorithmic idea:** *Pseudopolynomial algorithms*
  running time polynomial in input size **and** largest integer *value*

$\rightsquigarrow$ can be efficient if only small-integer inputs needed

- **Impossibility results:** strong NP-hardness

Loophole 2: Contrived worst cases

- **Loophole:** problem only hard on most contrived worst-case inputs
- **Algorithmic idea:** *Fixed-parameter algorithms*
  running time polynomial in input size **and** some other *parameter* of the input

$\rightsquigarrow$ can be efficient if only small-parameter inputs needed

- **Algorithmic idea 2:** *Efficient exponential algorithms*
  try to reduce base of exponential growth to delay combinatorial explosion (a bit)
- **Impossibility results:** $\mathsf{W}[P]$-hardness, $\mathsf{W}[1]$-hardness

Loophole 3: Exact answers

- **Loophole:** optimization problem only hard when precisely optimal solution requested
- **Algorithmic idea:** *Approximation algorithms*, PTAS, FPTAS
  polytime through compromise on quality
  but with a provalbe guarantee on how much worse results get!
- **Impossibility results:** hardness of approximation, APX-hardness

Loophole 4: Rare Bad Cases

- **Loophole:** problem is hard in unpredictable ways if algorithms take "unlucky turns"
- **Algorithmic idea:** *Randomized algorithms*
  introduce controlled randomness to escape local bad luck
- **Techniques:** *Average-case analysis* and *smoothed analysis*
  can tell whether randomization could be promising
- **Impossibility results:** randomized complexity classes (ZPP, RP, PP)

Overview of the module

Goals:

- enhance your toolbox of algorithmic methods and techniques
  $\rightsquigarrow$ here: focus on hard problems
- enhance your toolbox of hardness and lower-bounding techniques
- focus on algorithm theory and rigorous methods $\rightsquigarrow$ formal proofs, mathematical machine models, asymptotic analysis

Overview of the module

Goals:

- enhance your toolbox of algorithmic methods and techniques
  - $\rightsquigarrow$ here: focus on hard problems
- enhance your toolbox of hardness and lower-bounding techniques
- focus on algorithm theory and rigorous methods $\rightsquigarrow$ formal proofs, mathematical machine models, asymptotic analysis

Units: (preliminary plan)

0. Administrativa
1. Motivation & Outline
2. Complexity Theory Recap
3. Pseudopolynomial Algorithms
4. Fixed-parameter algorithms
5. Parameterized hardness
6. Advanced Parameterized Concepts
7. Randomization Basics
8. Randomized Complexity
9. Random tricks
10. Advanced Randomized Algorithms
11. Approximation Algorithms
12. Linear Programming
13. Inapproximability