Looking Ahead and Course Review

CSC 463

April 3, 2020

Why is P versus NP hard?

- We believe that not every problem in **NP** has a polynomial-time algorithm.
- The techniques that complexity theorists have developed have not worked to separate **P** and **NP**.
- Some of these techniques are discussed in Sipser's book (Chapter 9 and 10).

Diagonalization

- Recall that we used Cantor's diagonalization argument to show that there is a semi-decidable problem that is not decidable. So we can do something similar to show that there is a problem in **NP** not in **P**?
- The answer is no. This concept is made rigorous by the concept of **relativization.**

Theorem (Baker-Gill-Solovay (1975))

There are oracle Turing machines A, B *where* $P^A = NP^A$ *and* $P^B \neq NP^B$.

- This rules out a large class of possible proof techniques towards the P versus NP problem.

Circuit Complexity

- Another natural model of computation is **circuits.**
- Every problem in **P** has a family of circuits with $poly(n)$ gates.
- Hence if we can prove that a problem in **NP** cannot be solved using a family of polynomially-sized circuits, then we will have proven **P** $\neq$ **NP.**
- We know that there are Boolean functions f requiring $\Omega(\frac{2^n}{n})$-sized circuits to compute, but we do not know that these functions f define languages in **NP**.
- Boolean functions $f : \{0,1\}^* \mapsto \{0,1\}$ and languages are related by considering the languages $L = f^{-1}(0)$ or $L = f^{-1}(1)$.

Circuit Complexity

- Unfortunately this approach has not seemed to give strong separations in general so far for the **P** versus **NP** problem.
- We still cannot prove superlinear circuit size lower bounds functions in **NP**.

Theorem (Schnorr 1974)

Any family of circuits for the parity function $f(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i \pmod 2$ *requires at least* $3n - o(n)$ *gates.*

- We are still far from proving **P** $\neq$ **NP** with this approach!

Circuit Complexity

- There are more powerful results for **restricted** forms of circuits, some even for **NP**-Complete problems.

Theorem (Razborov 1985)

Let $k \leq n^{1/4}$. *Any* **monotone circuit** *for computing the clique function* $CLIQUE_{k,n}(x_1, \ldots, x_{\binom{n}{2}})$ *requires size* $2^{\epsilon\sqrt{k}}$.

In particular, this is not polynomially sized for $k = \Theta(n^{1/4})$.

- But it has been shown that these techniques used to prove these bounds for restricted circuits cannot work in general if we believe cryptography is possible in this universe. This observation is known as **the natural proofs barrier.**

Other computational models

- Complexity theorists have studied other computational models as a way to gain insight on the **P** versus **NP** problem.
- Interactive proofs can be considered as a generalization of **NP** where a verifier is probabilistic rather than a deterministic machine.
- The study of interactive proofs has to lead to important discoveries such as the **PCP Theorem** on probabilistically checkable proofs.
- The PCP Theorem then leads to guarantees on how well **NP**-hard optimization problems can be approximated in polynomial time under $\mathbf{P} \neq \mathbf{NP}$.

Other computational models

- Some examples of other computational models include:
  - Communication complexity.
  - Algebraic circuits for polynomials.
  - Computational with real numbers (Blum-Shub-Smale model)
  - Quantum computers and other physics-inspired models of computation.
  - Descriptive complexity and models inspired by logic.
  - Many more that have yet to be discovered!
- The hope is studying these other computational models will give insight on the nature of computation and find applications in other areas of computing, mathematics, and science.

Course Summary: Computability

- We define precisely what it means for a problem to be solvable by an algorithm using the Turing machine model.
- We gave some evidence for the Church-Turing thesis that any physically implementable model of computation can be efficiently simulated by a Turing machine.
- We showed some examples of **uncomputable** problems: the halting problem, Post Correspondence Problem, Kolmogorov-incompressibility of strings.
- The main techniques we used were diagonalization arguments and computable mapping reductions.
- This is related to formal program verification (Rice's Theorem) and foundations of mathematics (incompleteness theorems).

Course Summary: Time Complexity

- We also care about problems who are a decidable in principle, but may not have an **efficient** algorithm.
- We use polynomial-time algorithms as a definition of an efficient algorithm.
- Problems in **NP** are the problems where given a potential solution, we can verify if it is an actual solution efficiently. There are many practically relevant problems in **NP.**
- The concept of **NP**-Completeness is used to give evidence that problems do not have polynomial time algorithms.
- If $\mathbf{P} \neq \mathbf{NP}$, then any $\mathbf{NP}-$Complete problem does not have a polynomial time algorithm.

Course Summary: Time Complexity

- We proved the Cook-Levin theorem stating that Boolean satisfiability is a problem that is **NP**-Complete.
- From that result, we used a chain of polynomial time reductions to show that other problems are also **NP**-Complete: clique finding, independent set, vertex covering, graph colouring, Hamiltonian path etc.
- When **NP**-Complete problems are encountered in real life, approximation algorithms are often used to tackle them so that useful solutions can be obtained efficiently.

Course Summary: Space Complexity

- Another useful resource is memory or space.
- Space is a powerful resource. We expect **PSPACE** $\neq$ **P.**
- An example of a **PSPACE**-Complete problem is quantified Boolean satisfiability under polynomial time reductions.
- Proving **PSPACE**-Completeness is stronger evidence of intractibility than **NP**-Completeness, we conjecture all inclusions

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXP}$$

  are strict.
- We know for sure **P** $\neq$ **EXP** using the time hierarchy theorem.

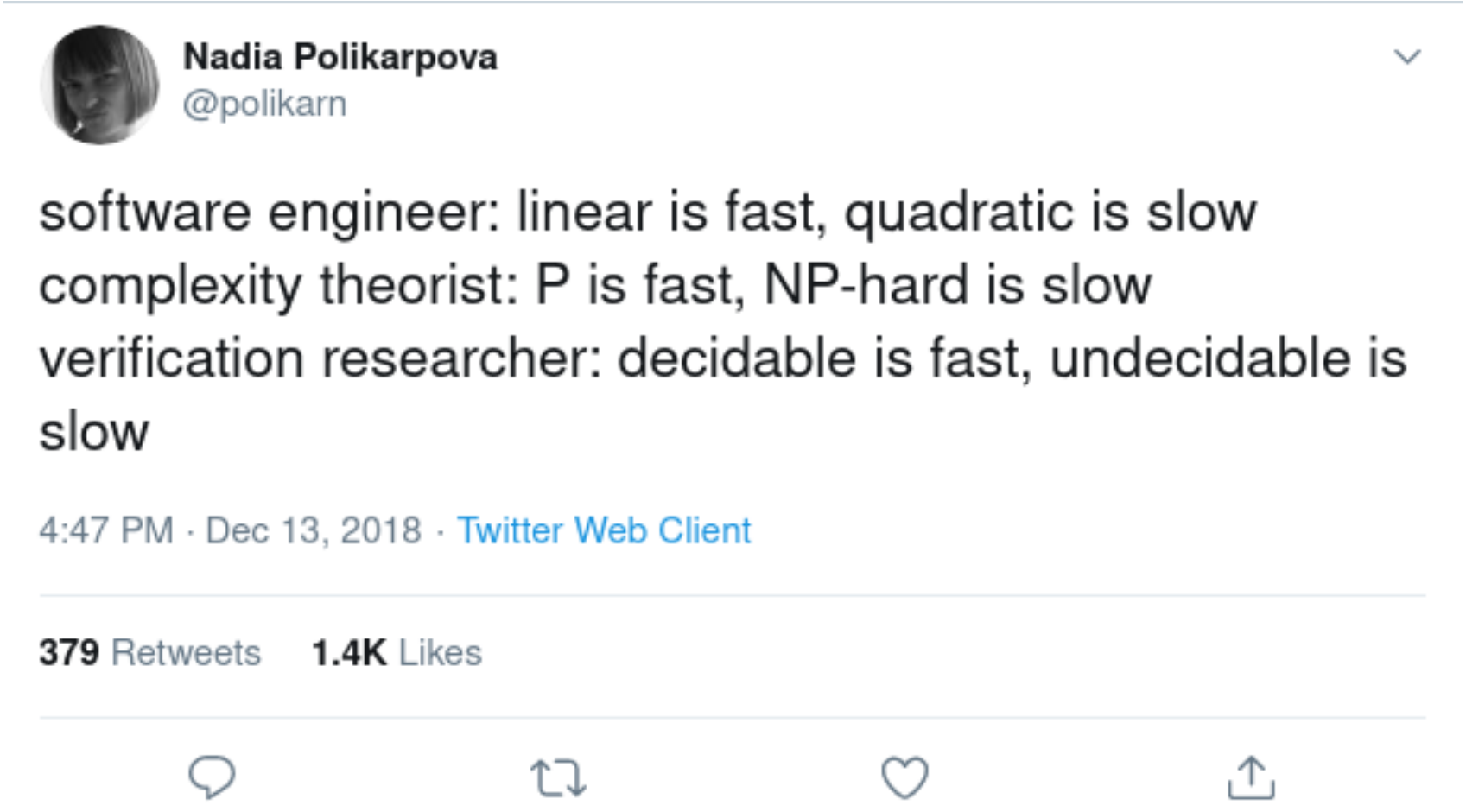
Course Summary: Space Complexity

- For some applications, we only have access to small space. This motivates the study of small space complexity classes **L** and **NL**.
- Determining if a **directed** graph has a path between two vertices s, t is a problem that is **NL**-Complete using log space reductions.
- We know the inclusions

$$\mathbf{L} \subseteq \mathbf{NL} = \mathbf{coNL} \subseteq \mathbf{P}$$

and expect all inclusions to the strict.

Course Summary



Course Summary

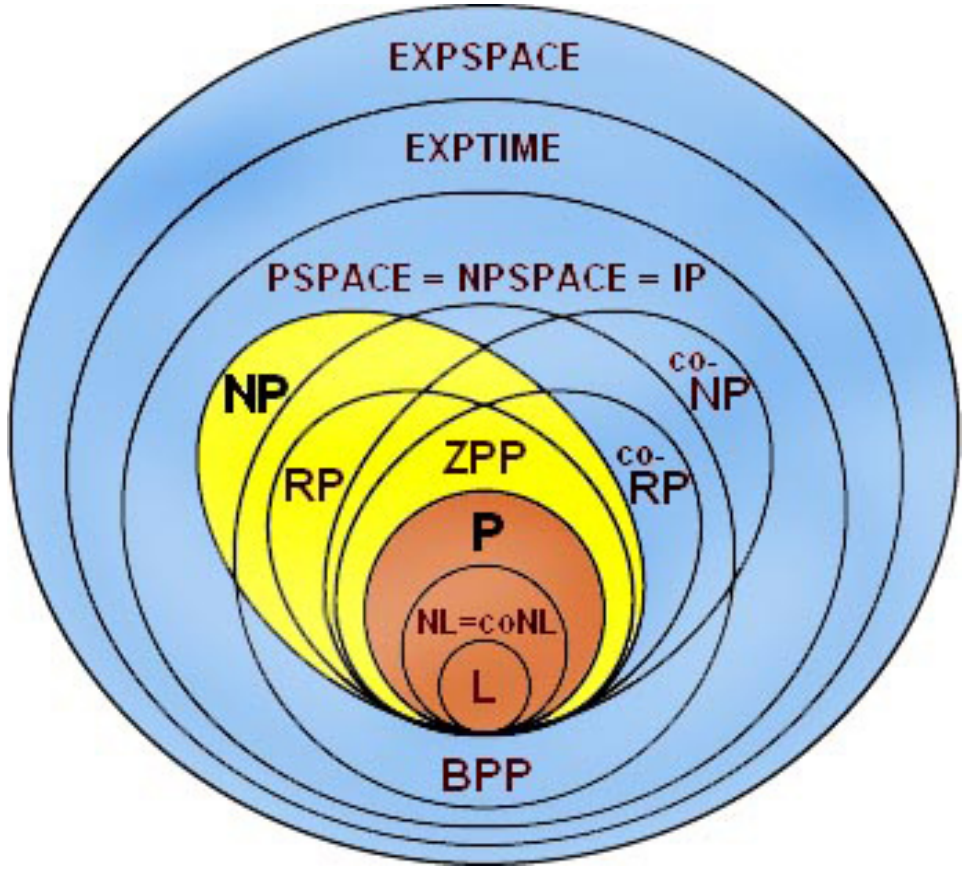


Figure: Image from
`https://jeremykun.com/2012/02/29/other-complexity-classes/`