Proof, Verification and **NP**-hardness

Ziyi Cai

2022/11/14

Review: Deterministic Time & Space Complexity

- The class $\mathbf{TIME}\big(T(n)\big)$ or $\mathbf{DTIME}\big(T(n)\big)$
  - A language L is in $\mathbf{TIME}\big(T(n)\big)$ iff there exists a TM $\mathbb{M}$ that runs in $cT(n)$ time and decides L.
- The class $\mathbf{SPACE}\big(S(n)\big)$ or $\mathbf{DSPACE}\big(S(n)\big)$
  - A language L is in $\mathbf{SPACE}\big(S(n)\big)$ iff there exists a TM $\mathbb{M}$ that runs in $cS(n)$ space and decides L.
- Time complexity classes
  - $\mathbf{P} := \bigcup_{i=1}^{\infty} \mathbf{TIME}\big(n^i\big)$
  - $\mathbf{EXP} := \bigcup_{i=1}^{\infty} \mathbf{TIME}\left(2^{n^i}\right)$
- Space complexity classes
  - $\mathbf{L} := \mathbf{SPACE}(\log n)$
  - $\mathbf{PSPACE} := \bigcup_{i=1}^{\infty} \mathbf{SPACE}\big(n^i\big)$
  - $\mathbf{EXPSPACE} := \bigcup_{i=0}^{\infty} \mathbf{SPACE}\left(2^{n^i}\right)$
- $\mathbf{L} \subseteq \mathbf{P} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXP}$

NP: a view from proof and verification

Efficiently Verifiable Solutions

- Sudoku
  - hard to solve a puzzle
  - easy to check a solution

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

The Class **NP**: Proof and Verification

- A language L is in $\mathbf{NP}$ if there exists a polynomial $p: \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM $\mathbb{V}$ such that the following conditions hold:
  - **Completeness**: $\forall x \in L, \exists y \in \{0,1\}^{p(|x|)}$ such that $\mathbb{V}(x,y) = 1$.
  - **Soundness**: $\forall x \notin L, \forall y \in \{0,1\}^{p(|x|)}$ we have $\mathbb{V}(x,y) = 0$.
- $\mathbb{V}$ is called *verifier*.
- If $x \in L$ and $y \in \{0,1\}^{p(|x|)}$ satisfy $\mathbb{V}(x,y) = 1$, then we call y a *certificate* (or *witness*) for x (with respect to the language L and the machine $\mathbb{V}$).
- $\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{EXP}$

Stephen Cook

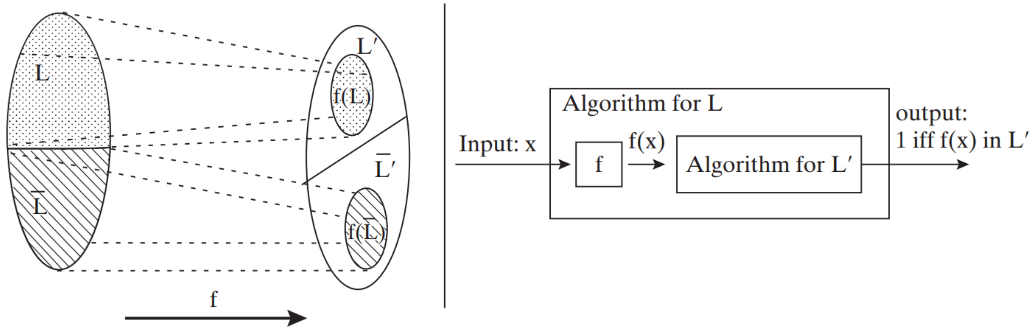
Leonid Levin

Examples of **NP** languages

- **VERTEX-COVER**: Given a graph G and $k \in \mathbb{N}$, decide whether G has a vertex cover of size k.
  - The certificate is a vertex cover of size k.
- **$0/1$ INTEGER PROGRAMS**: Given a set of linear inequalities with rational coefficients over variables $u_1, \ldots, u_n$, decide whether there is an assignment of numbers in $\{0,1\}$ to $u_1, \ldots, u_n$ that satisfies it.
  - The certificate is a satisfying assignment.
- **SAT**: $\mathrm{SAT} \coloneqq \{\langle \phi \rangle : \phi \text{ is a satisfiable CNF}\}$
  - The most important problem in theoretical computer science.
  - The certificate is a satisfying assignment.

Karp-Reduction and **NP**-completeness

- A language $L \subseteq \{0,1\}^*$ is *polynomial-time Karp reducible* to a language $L' \subseteq \{0,1\}^*$, denoted by $L \leq_p L'$, if there is a polynomial-time computable function $f: \{0,1\}^* \to \{0,1\}^*$ such that for every $x \in \{0,1\}^*, x \in L \Leftrightarrow f(x) \in L'$.
- A language L is **NP**-hard if $L' \leq_p L$ for all $L' \in \mathbf{NP}$.
- A language L is **NP**-complete if $L \in \mathbf{NP}$ and L is **NP**-hard.

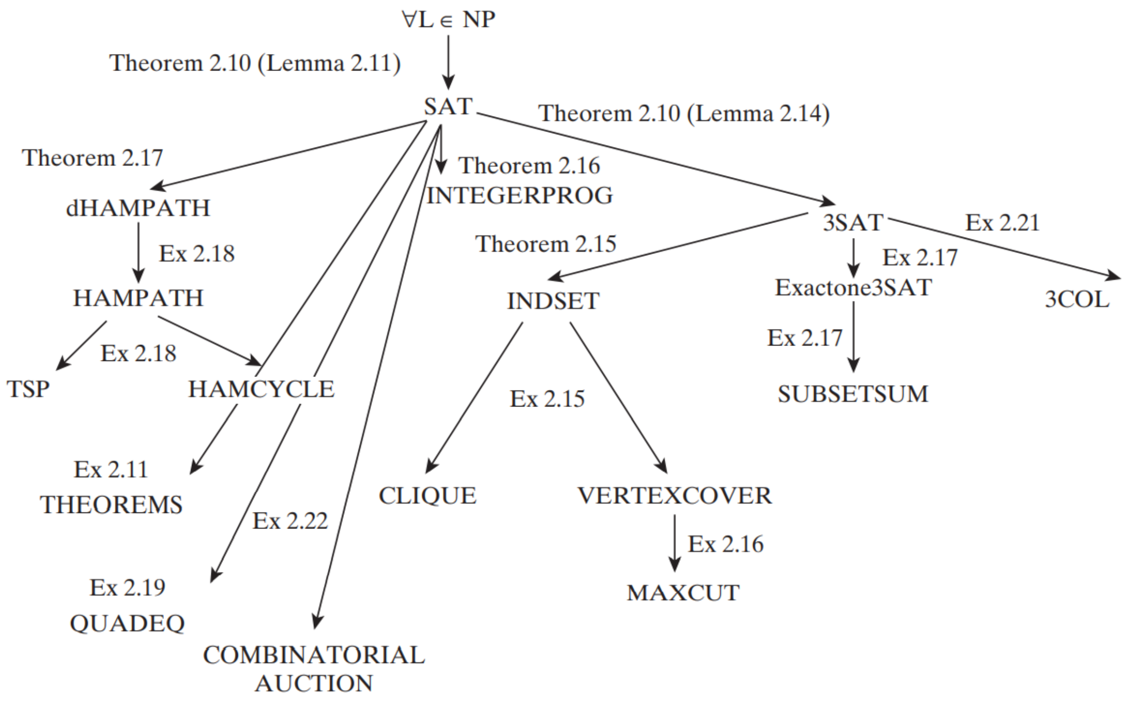


Arora and Barak (2009). Computational Complexity: A Modern Approach, Cambridge University Press. p. 43.

The Web of Reductions

Cook-Levin Theorem. (Cook 1971, Levin 1973)
SAT is NP-complete.

- Proof idea of Cook-Levin Theorem:
  - The computation of the verifier can be formulated by a polynomial-size CNF.
- Another example: $0/1$ IPROG is $\mathbf{NP}$-complete.
  - $u_1 \vee \overline{u_2} \vee \overline{u_3}$ can be expressed as $u_1 + (1 - u_2) + (1 - u_3) \geq 1$
- Why do complexity theorists love reductions?
  - Human creativity is more adaptable to algorithm-design than to proving lower bounds.



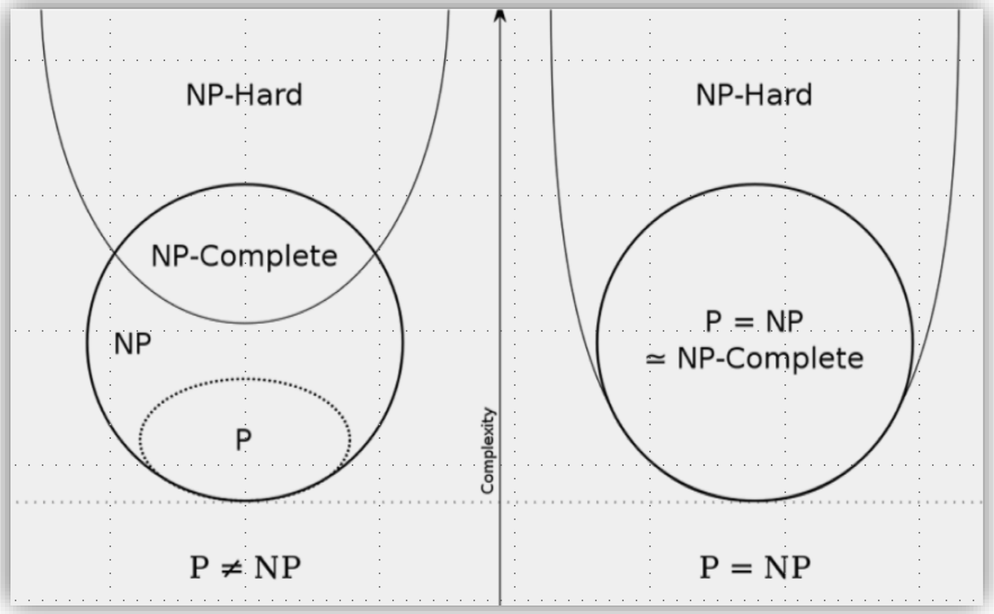
Arora and Barak (2009). Computational Complexity: A Modern Approach, Cambridge University Press. p. 51.

P vs. NP

- Is checking the correctness of a proof harder than presenting a proof?
  - $\mathrm{THEOREMS} \coloneqq \{\langle \varphi, 1^n \rangle : \varphi \text{ has a formal proof of length} \leq n \text{ in system } \mathcal{A}\}$ is **NP**-complete.
- $\mathbf{P} = \mathbf{NP}$ doesn't imply that we can find algorithms for **NP** problems efficiently.
  - There are problems in **P** which take thousands of years to solve.
    - PRIMES is in **P**, proved by Agrawal and Kayal in 2002.
- If $\mathbf{P} \neq \mathbf{NP}$, then there exists **NP** intermediate languages!

  Ladner's Theorem. (Ladner 1975)
  Suppose that $\mathrm{P} \neq \mathrm{NP}$. Then there exists a language $L \in \mathrm{NP} \backslash \mathrm{P}$ that is not NP-complete.

  - Candidates for **NP** intermediate languages: integer factorization, graph isomorphism.
  - They are neither known to be in **P** nor known to be **NP**-complete.



What is a theorem-proving procedure?

- An alternative definition of $\mathbf{NP}$: $L \in \mathbf{NP}$ if and only if there exists TM $\mathbb{V}$ such that
  - $\mathbb{V}$ runs in polynomial time
  - **Completeness**: $\forall x \in L, \exists \mathbb{P}, \langle \mathbb{P}, \mathbb{V} \rangle(x) = 1$.
  - **Soundness**: $\forall x \notin L, \forall \mathbb{P}, \langle \mathbb{P}, \mathbb{V} \rangle(x) = 0$.
  - *There are no limits on the computational power of* $\mathbb{P}$.
- A probabilistic version of proving?
  - $\mathbb{V}$ runs in polynomial time
  - **Completeness**: $\forall x \in L, \exists \mathbb{P}, \Pr[\langle \mathbb{P}, \mathbb{V} \rangle(x) = 1] \geq 2/3$.
  - **Soundness**: $\forall x \notin L, \forall \mathbb{P}, \Pr[\langle \mathbb{P}, \mathbb{V} \rangle(x) = 0] \leq 1/3$.
  - *There are no limits on the computational power of* $\mathbb{P}$.
- What if interaction between prover and verifier is allowed?

Interactive Proof: An Intuitive Example

- Bob has one red sock and one green sock.
- How can he convince A, who is color blind, that the socks are of different color?
- Hint: a private coin is needed.
- Interactive proof is powerful.
  - GNI $\in$ **IP**, but it is not likely to be in **NP**.

Theorem. (Shamir 1990)
IP = PSPACE.

Coping with **NP**-completeness

Is randomness helpful?

- A language L is in $\mathbf{BPP}$(Bounded-error Probabilistic Polynomial Time) if there exists a polynomial-time TM M and a polynomial $p: \mathbb{N} \to \mathbb{N}$ such that for every $x \in \{0,1\}^*$

$$\Pr_{r \in_R \{0,1\}^{p(|x|)}}[M(x,r) = L(x)] \geq \frac{2}{3}.$$

- Example: Polynomial Identity Testing
- Clearly, $\mathbf{P} \subseteq \mathbf{BPP}$.
- Randomness is costly.
  - We cannot achieve true randomness in real world.
  - The difficulty of generating random bits depends on the number and quality of random bits.
  - Randomness can be viewed as a resource, like space and time.

- Derandomization:
  - Can we use fewer random bits?
  - Can we use weaker random sources instead of i.i.d. random variables?
  - Can we replace the probabilistic algorithm with a deterministic one without a significant loss of efficiency?
- Most researchers believe that $\mathbf{P} = \mathbf{BPP}$.
- However, it remains open.
  - Full derandomization of $\mathbf{BPP}$ will imply some lower bounds in circuit complexity, which are difficult to prove.

Is non-optimality OK?

- An approximation algorithm $\mathcal{A}(G)$ for minimum vertex cover of graph G:
  - Start with $S = \emptyset$.
  - Whenever an edge (u, v) is not covered, we join u, v into S.
- Define the approximation ratio $\alpha(\mathcal{A}) \coloneqq \max_{G} \frac{\mathcal{A}(G)}{\mathrm{MVC}(G)}$.
- $\alpha(\mathcal{A}) \leq 2$
  - Any vertex cover of the input graph must use a distinct vertex to cover each edge that was considered in the process.

- hardness of approximation

Theorem. (Dinur and Safra, 2005)
MVC cannot be approximated within a factor of 1.3606 for any sufficiently large vertex degree unless $\mathrm{P} = \mathrm{NP}$.

Parameterized Algorithm

- A naïve algorithm for k-**VC**:
  - Enumerate and check all subsets $S \subseteq V$ with $|S| \leq k$.
  - The time complexity is roughly $O(|G|^k)$.

- Another naïve algorithm $\mathrm{DFS}(G, k)$:
  - If $k < 0$ then reject
  - If G has no edge then accept
  - Else choose an arbitrary edge (u, v) in G
  - Accept if and only if $\mathrm{DFS}(G - u, k - 1)$ accepts or $\mathrm{DFS}(G - v, k - 1)$ accepts.
- This recursive algorithm decreases k by one and branches to two after each recursion.
- The time complexity is roughly $O\left(2^k \cdot |G|\right)$.

A problem is fixed parameter tractable (FPT) with respect to k if there exists a TM $\mathbb{M}$ that runs in $f(k) \cdot |x|^{O(1)}$ time for every $x \in \{0,1\}^*$, where f is a computable function of k which is independent of n.

Average-case Complexity

- Many algorithms work well on "average" cases.
  - Quicksort (non-randomized) runs worst when the input is nearly sorted, which is uncommon in the real world.
  - 3-**COLOR** can be solved in linear time with high probability on $G(n, 1/2)$.
- What is the class of distributions that makes sense "in practice"?
  - $\mathbf{P}$-samplable distribution
- A distribution problem is a pair $\langle L, \mathcal{D} \rangle$, where
  - L is a language
  - $\mathcal{D} = \{\mathcal{D}_n\}$ is a sequence of distributions over $\{0,1\}^n$.
- The class $\mathrm{dist}\mathbf{P}$ is the average-case analog of $\mathbf{P}$.
- $\langle L, \mathcal{D} \rangle \in \mathrm{samp}\mathbf{NP}$ if and only if $L \in \mathbf{NP}$ and $\mathcal{D}$ is $\mathbf{P}$-samplable.
- $\mathrm{samp}\mathbf{NP} \subseteq \mathrm{dist}\mathbf{P}$?
  - Are $\mathbf{NP}$–hard problems efficiently solvable most of the time?
- Impagliazzo's five worlds of complexity.
  - *Algorithmica*: $\mathbf{P} = \mathbf{NP}$, a computational utopia, no **provably** secure cryptographic scheme
  - *Heuristica*: $\mathbf{P} \neq \mathbf{NP}$, $\mathrm{samp}\mathbf{NP} \subseteq \mathrm{dist}\mathbf{P}$, similar to *Algorithmica*
  - *Pessiland*: $\mathrm{samp}\mathbf{NP} \not\subseteq \mathrm{dist}\mathbf{P}$, but still no *one-way function* exists, worst-possible world
  - *Minicrypt* and *Cryptomania*: worlds of cryptography
- Which world do we live in?

Complexity theory is hard.

- Complexity is about some internal properties of problems.
- We can now only classify problems but have no idea about the reason that makes them computationally hard.
- Most results are *relative*, which means relating one fact regarding computational complexity to another.
  - *Absolute* results of "big questions" are rare.
- Some proof techniques are shown to be restricted.
  - relativizing barrier, natural proof
- However, a single new "unnatural" technique will open the floodgates for a great many lower bounds.

If people do not believe that mathematics is simple, it is only because they do not realize how complicated life is.
- John von Neumann, 1947

Thanks for listening.