A Glance at *Computational Complexity*

TOWARDS FEASIBLE AND EFFICIENT COMPUTATION

Xinyu Mao

11/13/2021

Notations

- $\mathbb{N} := \{0,1,2,3, \dots\}$.
- The unary string $\underbrace{111 \cdots 1}_{n \text{ times}}$ is denoted by 1^n.
- For a TM $\mathbb{M}$, define $\mathbb{M}(x) :=$ the output of $\mathbb{M}$ on input x.
- For a *language* L, define its *indicator function* $\mathbf{1}_L(x) := \begin{cases} 1, \text{if } x \in L; \\ 0, \text{if } x \notin L. \end{cases}$
- $\llcorner \cdot \lrcorner$ means binary representation(or *encoding*)
  - e.g. $\llcorner \mathbb{M} \lrcorner$, $\llcorner i \lrcorner$, $\llcorner \phi \lrcorner$.

The notion of complexity

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

Running time

- Let $T : \mathbb{N} \to \mathbb{N}$ be a function.
- *We say $\mathbb{M}$ computes f in $T(n)$ time if for all input x, its computation requires are* ***most*** *$T(|x|)$ steps.*
- *Definition.* T is *time constructable if $T(n) \geq n$ and there is a TM $\mathbb{M}$ that computes the function $1^n \mapsto \llcorner T(n) \lrcorner$ in time $T(n)$.*
  - *Motivation: we often need a timer when doing simulation.*
  - *Common time functions such as $cn, n^2, 2^n$ are time-constructable.*

Universal Turing machine, revisited

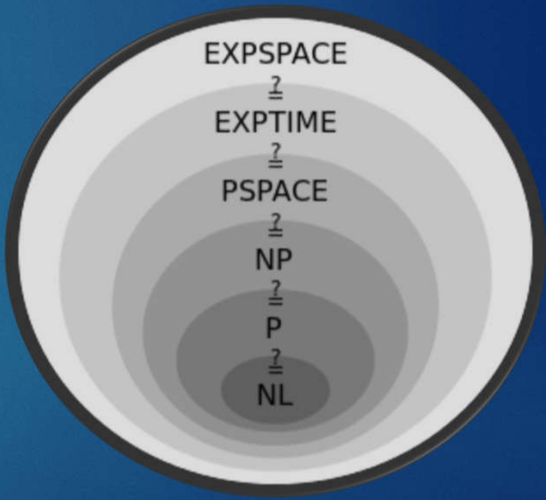
- We assume that $T(n)$ is time constructable.
- *Theorem. if f is computable in time $T(n)$ by a TM $\mathbb{M}$ using alphabet Γ, then it is computable in $4\, log|\Gamma|\; T(n)$ time by a TM $\widetilde{\mathbb{M}}$ using alphabet* $\{0,1,\triangleleft,\square\}$.
  - The alphabet does not matter.
- *Theorem. A k-tape TM with running time $T(n)$ can be simulated by a single-tape TM within $O(T(n)^2)$ time.*
  - *The number of tapes does not matter.*
- *Theorem. There exists a universal TM $\mathbb{U}$ such that $\mathbb{U}(\alpha, x)$ halts within $c_\alpha T^2$ steps if $\mathbb{M}_\alpha$ halts within T steps.*
  - *c_α is a constant depending on $\mathbb{M}_\alpha$.*
  - More efficient emulation: $c_\alpha T \log T$. (Hennie and Stearns [HS66])

Does computational model matters?

- **Church-Turing thesis**: When it comes to *computability*, it suffices to study Turing machines.
- The intrinsic complexity of problems should not depend on specific computational model.
- **Cobham-Edmonds thesis**: Every physically realizable computation model can be simulated by a TM with polynomial overread.
- Summary: About Turing machines
  - The alphabet can be reduced to $\{0,1,\triangleleft,\square\}$.
  - The number of tapes does not matter.
  - UTM is efficient.

Time and space resources: The class $\mathbf{P}$ and $\mathbf{PSPACE}$

- *The class* $\mathbf{DTIME}(T(n))$
  - *A language L is in $\mathbf{DTIME}(T(n))$ iff there exists a TM $\mathbb{M}$ that runs in $cT(n)$ time and decides L.*
- $\mathbf{P} \coloneqq \bigcup_{i=1}^{\infty} \mathbf{DTIME}(n^i)$, $\mathbf{EXP} \coloneqq \bigcup_{i=1}^{\infty} \mathbf{DTIME}\left(2^{n^i}\right)$.
  - Why not define $\mathbf{EXP}$ as $\bigcup_{i=1}^{\infty} \mathbf{DTIME}(2^{in})$?
- The class $\mathbf{SPACE}(S(n))$
  - A language L is in $\mathbf{SPACE}(S(n))$ iff there exists a TM $\mathbb{M}$ that runs in $cS(n)$ space (number of locations ever visited on the word tapes) and decides L.
- $\mathbf{PSPACE} \coloneqq \bigcup_{i=1}^{\infty} \mathbf{SPACE}(n^i)$.
- $\mathbf{P} \subseteq \mathbf{PSPACE}$.



P, NP and P vs. NP

Still open.

$\mathbf{P}$ vs. $\mathbf{NP}$: Proving vs. Verifying

- *Definition. A language L is in $\mathbf{NP}$ if there exists a polynomial $p: \mathbb{N} \to \mathbb{N}$ and a $\mathbf{P}$-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**: $\forall\, x \notin L, \mathbb{V}(x, y) = 0$ holds for all y.*
  - *$\mathbb{V}$ is called verifier, and y with $\mathbb{V}(x, y) = 1$ is called the certificate of x.*
- Motivation: $\mathbf{NP}$ languages can be efficiently verified.
- Clearly, $\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{EXP}$.
- $\mathbf{P}$ vs. $\mathbf{NP}$: Is checking the correctness of a proof harder than presenting a proof?

Donald Knuth

I don't believe that the equality $\mathbf{P} = \mathbf{NP}$ will turn out to be helpful even if it is proved, because such a proof will almost surely be nonconstructive.
Donald Knuth

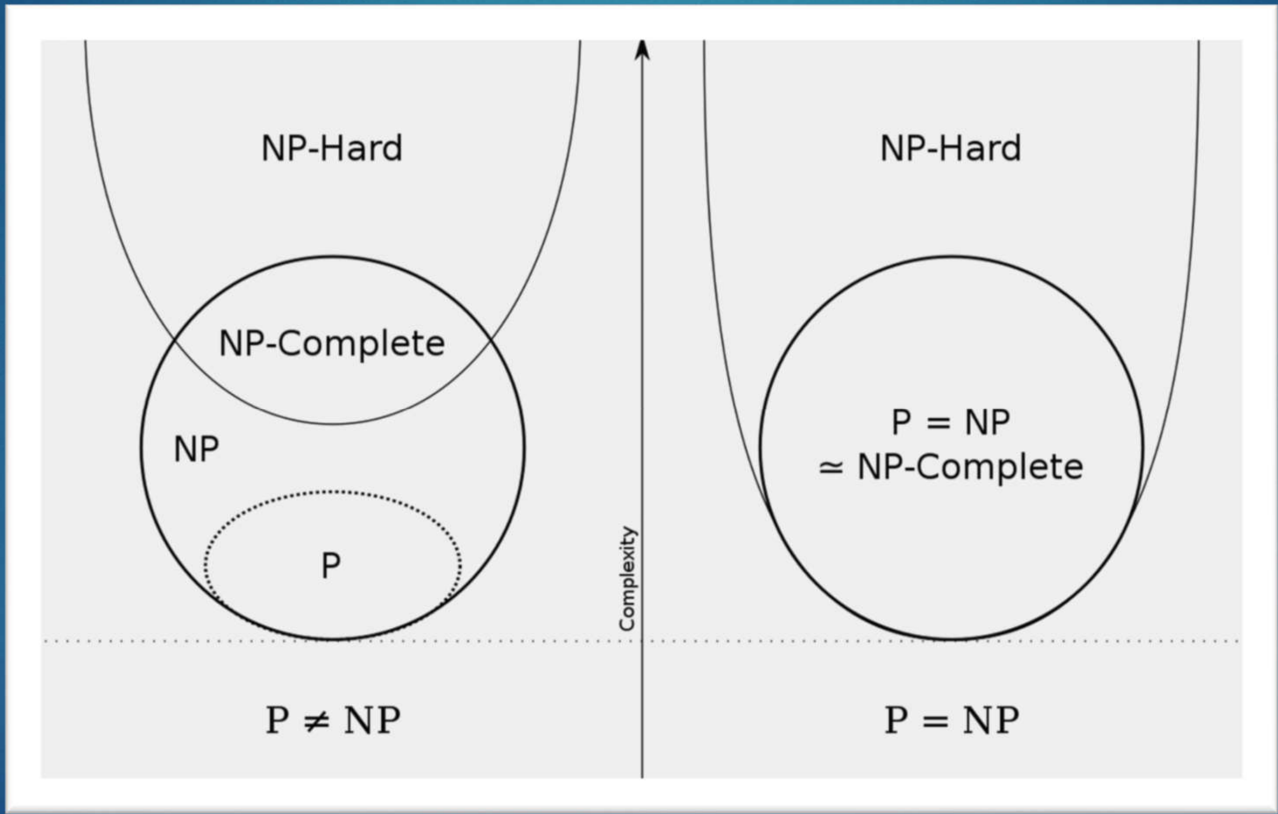
Examples of **NP** languages

- **SUBSET-SUM**: Given n numbers $A_1, \ldots, A_n$ and a number T , decide if there is a subset of the numbers that sums up to T .
  - The certificate is the list of members in such a subset.
- **VERTEX-COVER:** Given a graph G and $k \in \mathbb{N},$ decides whether G has a vertex cover of size k.
  - Vertex cover: a subset of vertices that 'covers' all edges.
  - The certificate is a vertex cover of size k.
- **SAT**: $\mathrm{SAT} \coloneqq \{\llcorner \phi \lrcorner : \phi$ is a satisfiable CNF$\}$.
  - Conjunction Normal Form
  - The certificate is the satisfying assignment.
- ……

Karp-Reduction and NP-completeness

- *Definition. Language L is polynomial-time Karp reducible to a language L' if there is a polynomial-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that for every $x \in \{0,1\}^*, x \in L$ if and only if $f(x) \in L'$.*
  - *Denoted by $L \leq_K L'$.*
- *A language L is NP-hard if $L' \leq_K L$ for all $L' \in \mathbf{NP}$.*
- *A language L is NP-complete if $L \in \mathbf{NP}$ and L is NP-hard.*
- *Theorem.* SAT *is NP-complete.*
  - Proof idea: the computation of the verifier can be formulated by a polynomial-size CNF.
- In fact, **SUBSET-SUM, VERTEX-COVER** are NP-complete as well.

NP-completeness and P vs. NP



Nondeterministic Turing Machine

- *An NDTM (Nondeterministic Turing Machine) has two transition functions δ_0, δ_1 and a special state denoted by q_{accept}.*
- Nondeterminism provides **the power of guessing**.
- *An NDTM $\mathcal{N}$ accepts x, denoted by $\mathcal{N}(x) = 1$, if there exists some sequence of choices that makes $\mathcal{N}$ reach q_{accept} on the input x.*
  - *Otherwise $\mathcal{N}$ refuses x, denoted by $\mathcal{N}(x) = 0$.*
- *We say that $\mathcal{N}$ runs in $T(n)$ time if for every input x and every sequence of nondeterministic choices, $\mathcal{N}$ reaches either q_{halt} or q_{accept} within $T(|x|)$ steps.*
- *The class* $\mathbf{NTIME}(T(n))$.
- *Theorem.* $\mathbf{NP} = \bigcup_{i=1}^{\infty} \mathbf{NTIME}(n^i)$.

In the face of hardness

The philosophers have only interpreted the world,
in various ways; the point is to change it.
Karl Marx

What can we do, if the world is so tough?

- We *solve a problem* in the following sense:
  - give the exact answer;
  - solve all cases;
  - use the same algorithm for all cases(uniformity);
- Relaxing the requirements
  - Allow errors on some instances(use *the power of randomness*)?
  - Give approximate answers(usually for counting problems)?
  - Not for all cases(*average-case complexity*)?
  - Choose different algorithms depending on the input(non-uniformity)?

The power of randomness

- *Definition. A language L is in* **BPP** *(Bounded-error Probabilistic Polynomial Time) if there exists a* $\mathbf{P}$*-time TM* $\mathbb{M}$ *and a polynomial* p *such that*

$$\Pr_{r \in_p \{0,1\}^{p(|x|)}} \left(\mathbb{M}(x,r) = \mathbf{1}_L(x)\right) \geq \frac{2}{3} \text{ for all } x \in \{0,1\}^*.$$

- Example: Primality test
  - **PRIMES** $:= \{\llcorner p \lrcorner : p \text{ is a prime}\}$.
  - Lehmann primality test → **PRIMES** $\in \mathbf{BPP}$
  - One-sided error primality test: Miller-Rabin primality test
  - Deterministic primality test [PRIMES is in P. Manindra Agrawal, Neeraj Kayal, Nitin Saxena]
- Clearly, $\mathbf{P} \subseteq \mathbf{BPP}$
- $\mathbf{BPP} = \mathbf{P}$? : the power of randomness is unknown.

Undirected Connectivity: the accidental tourist sees it all

- Let $G = (V, E)$ be a undirected graph, and $s, t \in V$
- Is there a path from s to t in G?
- $\text{UPATH} := \{\langle G, s, t\rangle : \text{there is a path from } s \text{ to } t\}$.
- Naive BFS: linear time, $\Omega(|V|)$ space.
- The random walk algorithm
  - *Start a simple random walk from s;*
  - *If the random walk reaches t within $6|V||E|$ steps, output **True**.*
  - $O(\log |V|)$ space.
  - One-sided error, success with probability $\geq \frac{2}{3}$.

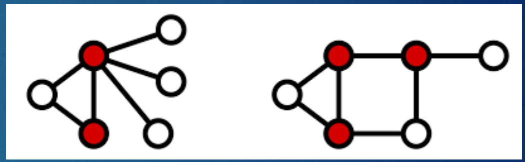
Lehmann primality test*

- *Input: Odd integer* N
- $\alpha_1, \alpha_2, \dots, \alpha_k \in_p \{1, 2, 3, \dots, N-1\}$
- *If* $\gcd(\alpha_i, N) > 1$, *output* ***COMPOSITE***
- *Compute* $\beta_i := \alpha_i^{\frac{N-1}{2}} \bmod N$
- *If* $(\beta_1, \beta_2, \dots, \beta_k) = (\pm 1, \pm 1, \dots, \pm 1)$ *but not all* β_i *equal to* 1
  - *Output* ***PRIME***
- *Output* ***COMPOSITE***

- We study the mapping
$$f: \mathbb{Z}_N^* \to \mathbb{Z}_N^*, x \mapsto x^{\frac{N-1}{2}}.$$
- If N is a prime, $f(x) \in \{-1, 1\}$ and
$$\Pr_{a \in_p \mathbb{Z}_N^*}(f(a) = 1) = \frac{1}{2}.$$
- If N is a composite number, then exactly one of the following happens:
  - $\Pr_{a \in_p \mathbb{Z}_N^*}(f(a) \notin \{-1, 1\}) \geq \frac{1}{2}$.
  - $f(x) = 1, \forall x \in \mathbb{Z}_N^*$.
- Lehmann primality test errs with probability $\leq \frac{1}{2^k}$.
- Tow-sided error.

Approximation: an example

- Minimum Vertex Cover of graph G.
- An approximation algorithm $\mathcal{A}(G)$:
  - *Start with $S = \emptyset$.*
  - *Whenever an edge (u, v) is not covered, we join u, v into S.*
- Define the *approximation ratio* $\alpha(\mathcal{A}) := \max_G \frac{\mathcal{A}(G)}{\mathrm{MVC}(G)}$.
- *Theorem.* $\alpha(\mathcal{A}) \leq 2$.
- Further studies in complexity: the hardness of approximation.



Average-case complexity

- Motivation: We may assume that the input obeys some (simple) distribution if solving the problem on all cases are way too hard.
- *Definition. A distributional problem is a pair $\langle L, \mathcal{D} \rangle$, where*
  - *L is a language;*
  - *$\mathcal{D} = \{\mathcal{D}_n\}$ is a sequence of distributions;*
  - *$\mathcal{D}_n$ is a distribution over $\{0,1\}^n$.*
- The class $\mathrm{dist}\mathbf{P}$.
- $\langle L, \mathcal{D} \rangle \in \mathrm{samp}\mathbf{NP}$ if $L \in \mathbf{NP}$ and $\mathcal{D}$ is a $\mathbf{P}$-samplable.

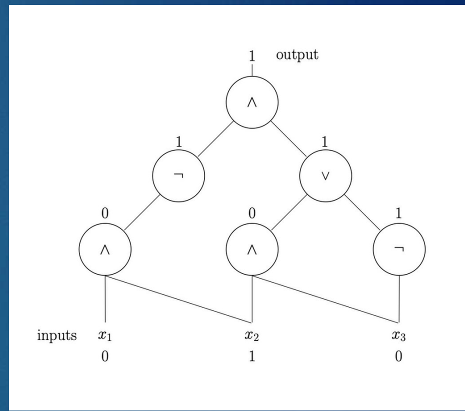
The average-case version P vs. NP

- $\text{samp}\mathbf{NP} \subseteq \text{dist}\mathbf{P}$?
  - Are NP-hard problems hard only in the worst cases, but easy most of the time?
  - Or, can we sample hard instances efficiently?
- Which world do we live in ? [Impagliazzo's five worlds]
  - Algorithmica: $\mathbf{P} = \mathbf{NP}$.
  - Heuistica: $\mathbf{P} \neq \mathbf{NP}$ but $\text{samp}\mathbf{NP} \subseteq \text{dist}\mathbf{P}$.
  - Pessiland: $\text{samp}\mathbf{NP} \not\subseteq \text{dist}\mathbf{P}$, but still there is no one-way function(we cannot even make use of hardness!).

Emm… I think we definitely live in Algorithmica or Heuisitca.

Non-uniform model: *Circuit Complexity*

- *Definition. Circuit family*
  - Motivation: design an algorithms for input with fixed length.
  - *The size of a circuit C, denoted by $|C|$, is the number of gates in C.*
  - *An $S(n)$-size circuit family is a sequence of circuits $\{C_n\}$, where C_n has n inputs, and $|C_n| \le S(n)$ for every n.*
- *$\{C_n\}$ accepts language L if $1_L(x) = C_{|x|}(x)$ for all $x \in \{0,1\}^*$.*
  - *For any unary language U, there exists some circuit accepting it.*
  - *Not all unary languages are **decidable**.*
  - The function $1^n \mapsto C_n$ can be uncomputable.



A circuit of size 6 with 3 inputs.

Non-uniformity is stronger than randomness

- *The class* $\mathbf{SIZE}(S(n))$.
- $\mathbf{P}_{/\mathrm{poly}} \coloneqq \bigcup_{i=1}^{\infty} \mathbf{SIZE}(n^i)$.
- *Theorem.* $\mathbf{BPP} \subseteq \mathbf{P}_{/\mathrm{poly}}$.
  - *Proof idea: for each input length, devise a circuit according to the good random string.*

Epilogue

Don't think twice, it's alright.
Bob Dylan

> "Intuition and concepts constitute... the elements of all our knowledge, so that neither concepts without an intuition in some way corresponding to them, nor intuition without concepts, can yield knowledge."

IMMANUEL KANT

Great idea: Definitions say it all.

We success when we are at the right level of abstraction.

The end of the tour

- The central question: *What makes some problems computationally hard and others easy?*
  - *We don't know much about it…*
  - *Our major success in complexity theory is classifying, just like the periodic table in chemistry.*
  - *Progresses are rare in terms of the essence of 'complexity'.*
- *Why are these 'natural and intuitive' questions so hard?*
  - *These questions reflect the raw and chaotic reality of life.*

Thanks for listening☺