# Intro to Meta-Complexity

Rahul Santhanam

(University of Oxford)

# Complexity Theory: Goals

- Complexity theory studies the possibilities and limits of efficient computation
  - Can be thought of as “fine-grained” version of computability theory, where we are concerned with resource requirements for computable problems
- Given a problem L of interest, we wish to design *efficient algorithms* for L, and if efficient algorithms do not exist, to prove *complexity lower bounds*
- While many algorithmic techniques are known, our knowledge of strong lower bound techniques is more limited
- Hence, rather than showing unconditional lower bounds on L, we are often satisfied with showing that L is equivalent in complexity to some other problem L’ that we believe to be hard

# Complexity Theory: Main Problems

- NP vs P: Does every problem in NP (i.e., with polynomial-size solutions that can be efficiently verified) be solved efficiently (i.e., in polynomial time)?
  - By the phenomenon of NP-completeness, this is equivalent to the polynomial-time solvability of specific problems such as SAT, Clique, Integer Linear Programming etc.
- PSPACE vs P: Can every problem in polynomial space be solved in polynomial time?
  - By the phenomenon of PSPACE-completeness, this is equivalent to the polynomial-time solvability of specific problems such as validity of quantified Boolean formulas (TQBF)

# Complexity Theory: Main Problems

- EXP vs SIZE(poly): Can every problem in exponential time be solved by polynomial size Boolean circuits?
  - We know by time hierarchy theorem that there are problems in EXP that are not in P, but showing lower bounds against polynomial size circuits seems much harder
  - If the answer is no, then by the theory of pseudo-random generators [NW94,IW97], [every problem in randomized polynomial time can be solved in deterministic sub-exponential time
- Why are complexity lower bounds so hard to prove?
  - Various complexity-theoretic barriers to success of known techniques for proving lower bounds, eg., the relativization barrier [BGS75] and the natural proofs barrier [RR97]

# Complexity Theory: Connections

- Logic: The area of *proof complexity* studies which tautologies have polynomial-size proofs in propositional proof systems such as Resolution and Frege
  - NP=coNP iff there is a propositional proof system such that all tautologies have polynomial-size proofs
- Learning: The theory of PAC-learning gives a complexity-theoretic framework in which to study the efficient learnability of concepts
  - If NP=P, then polynomial-size programs can be learned efficiently from their input-output behaviour
- Cryptography: Complexity-theoretic crypto reduces the security of cryptographic protocols such as key agreement and message authentication to the hardness of specific computational problems such as Factoring

# SAT and Friends

- Problems such as SAT are fundamental in complexity theory
  - SAT: Given a Boolean formula $\varphi$ on $n$ variables, is there a satisfying assignment to $\varphi$?
  - Bounded NTM Halting: Given the description <M> of a non-deterministic TM M, an input x, and a number t in unary, does M halt on x within t steps?
  - SAT and Bounded NTM Halting are NP-complete
- Such problems can be thought of as resource-bounded or finitary versions of uncomputable problems
  - SAT is a finitary version of the Non-Emptiness problem, which asks whether a given Turing machine M accepts any inputs
  - Bounded NTM Halting is a finitary version of the Halting problem

# White-Box Problems

- SAT and Bounded NTM Halting are problems about computation – given a formula or description of a machine, we are asked for information about properties of the formula or machine: non-emptiness, halting etc.
- Several other examples
  - TQBF (given a quantified Boolean formula, is it valid) is PSPACE-complete
  - #SAT (given a formula, count the number of satisfying assignments) is #P-complete
  - CVP (given a circuit $C$ and an input $x$, is $C(x) = 1$) is P-complete
- These are all examples of *white-box* problems: given the description of a computational object, answer questions about its behaviour

# Black-Box Problems

- A *black-box* problem is the inverse of a white-box problem. Instead of being given a computational object and asked questions about its behaviour, you are given information about the behaviour and asked about the computational object that produces the behaviour
- Example: Fundamental Problem of Learning, where you are given a set of pairs $(x_i,b_i)$, $i = 1..n$, where each $x_i$ is of length $n$, and each $b_i$ a bit, as well as an integer $s$, and asked if there is a Boolean circuit $C$ of size at most $s$ such that $C(x_i) = b_i$ for each $i$

# Black Box, White Box

- In general, white box problems are fairly well understood in terms of their complexity relationships
- Black box problems are much more mysterious
- The complexity of black box problems is especially relevant to applications in logic, learning and cryptography

# Meta-Complexity

- Meta-complexity is the study of computational problems that are themselves about complexity, eg., the Minimum Circuit Size Problem (MCSP) or the problem of computing Kolmogorov complexity
- Meta-complexity as a topic: Which complexity classes do various meta-complexity problems lie in? What complexity lower bounds can we show for them? What reductions exist between them?
- Meta-complexity as a tool: Use meta-complexity to attack fundamental questions in computational complexity, learning theory, cryptography and proof complexity

# MCSP (Minimum Circuit Size Problem)

- MCSP: Given the truth table of a Boolean function F, and a parameter s, does F have Boolean circuits of size s?
- MCSP[s]: Given the truth table of a Boolean function F on log(N) variables, does F have Boolean circuits of size s(N)?
- MCSP is an example of a black-box problem: we are given an explicit representation of a Boolean function (i.e., its truth table), and want to know if it has a succinct description using circuits
- By interpreting a string as the truth table of a Boolean function, we can think of circuit size as a *complexity measure* of a string, and MCSP is a problem naturally associated with this measure

# MCSP and Complexity Lower Bounds

- Showing that $DTIME(2^{O(n)})$ does not have Boolean circuits of size $s(n)$ is equivalent to efficiently constructing NO instances of $MCSP[s(\log(N))]$ of size $N$ given input $1^N$
  - In one direction, efficiently constructing NO instances gives a way of generating the truth table of a Boolean function without circuits of size $s(n)$ in time $2^{O(n)} = poly(N)$ (where $N = 2^n$)
  - In the other direction, if $L$ in $DTIME(2^{O(n)})$ does not have Boolean circuits of size $s(n)$, then we can efficiently generate the truth table of $L_n$ in time $2^{O(n)} = poly(N)$, and this truth table is a NO instance of $MCSP[s(\log(N))]$

# The Complexity of MCSP

- MCSP is in NP
  - Given the truth table y (of size N) of a Boolean function F, and a parameter s, we guess a circuit C for F of size s, and check that y is the truth table of the function computed by C, by running C on each input z of size log(N) and verifying that C(z) is consistent with y
- Question: Is MCSP in P?
  - Naïve algorithm incurs an exponential cost by running over all candidate circuits
  - No if one-way functions exist [GGM86, RR97, KC00]
- Question: Is MCSP NP-complete?
  - Recently Hirahara [H22] showed that a version called Partial-MCSP, where the input truth table has some "don't care" symbols, is NP-complete

# Variants of MCSP for Other Circuit Classes

- Given any circuit class C, we can define the problem C-MCSP, where the input is the truth table of a Boolean function F, and a parameter s, and the question is whether F has C-circuits of size s
- Intuitively, it seems that C-MCSP should be harder if C is a more powerful class, but this is not formally the case!
- It has been known for a long while that DNF-MCSP is NP-complete [M79], however NP-completeness of MCSP is still a major open question
- In general, we understand C-MCSP better the weaker C is

# Meta-Complexity Problems Based on Other Complexity Measures

- Circuit size can be thought of as a complexity measure on strings, and MCSP is the computational problem corresponding to this measure
- Similarly, we can consider other complexity measures and the computational problems corresponding to them
  - K: Kolmogorov complexity
  - KS: Space-bounded Kolmogorov complexity
  - $K^{poly}$: Poly-time bounded Kolmogorov complexity
- Just as SAT is a finitary version of the uncomputable problem Non-Emptiness, MCSP and KS and $K^{poly}$ are finitary versions of K

# Inherent Compressibility

- It is clear that some strings should be much compressible than others, eg., a string of N zeroes should be more compressible than a random string
- Explicit redundancies in strings, such as re-occurring patterns, are exploited in algorithms such as the Lempel-Ziv algorithm and its variants
- But there could be redundancy that is not based on repetition
  - Eg., consider the strings “3141592653” and “2718281828”
- *Kolmogorov complexity* yields a notion of inherent compressibility

# Kolmogorov Complexity

- Let $U$ be a fixed universal Turing machine
- For any string $x$ in $\Sigma^*$, $K(x)$ is min $\{|p| : U(p, \varepsilon) = x\}$
  - Given $y$ in $\Sigma^*$, $K(x|y)$ is min $\{|p| : U(p,y) = x\}$
- Intuitively, $K(x)$ is the size of the smallest program that produces x when run on the empty string
- Examples
  - $K(0^N) \leq \log(N) + O(1)$, since we can describe $0^N$ (in a way that makes sense to a computable de-compressor) by using $\log(N)$ bits to describe $N$ and $O(1)$ bits to describe a program that outputs $0^N$ given $N$
  - $K(\pi_N) \leq \log(N) + O(1)$, where $\pi_N$ is the string consisting of the first $N$ bits of $\pi$

# Basic Properties

- (1) For every $x$ in $\Sigma^*$, $K(x) \leq |x| + O(1)$
  - Any string $x$ can be described by itself together with a program p of constant size that just prints $x$ out
- (2) For each integer $n$, there is $x$ of length $n$ such that $K(x) \geq n$
  - Straightforward counting argument
  - For any $i$, there are at most $2^i$ strings of Kolmogorov complexity $i$ (since there are at most $2^i$ descriptions of length $i$)
  - So there are at most $2^n-1$ strings of Kolmogorov complexity $< n$
  - By pigeonhole principle, there is a string $x$ of length $n$ with $K(x) \geq n$

# Meta-Complexity of K

- MKP: Input is a string x together with a parameter s, question is whether $K(x) \leq s$
- K: Given a string x, compute the Kt complexity of x
- MKP and K are uncomputable
  - Note that the problems reduce to each other in polynomial time, hence it is sufficient to consider one of them when analyzing complexity

# Uncomputability of Kolmogorov Complexity

- Suppose, for the sake of contradiction, that there is a TM $M$ that computes $K$
- Define a TM $N$ that accepts $x$ iff $K(x) \geq n$
  - By Basic Property (2) of $K$ complexity, $N$ accepts at least one string for each input length $n$
- Now define a sequence of strings $\{x_n\}$, $|x_n|=n$, as follows
  - For each $n$, $x_n$ is the lexicographically first string of length $n$ that $N$ accepts
  - Note that we can compute $x_n$ given $n$ by simulating $N$ on strings of length $n$ in lex order and outputting the first such string it accepts
  - This implies that $K(x_n) \leq \log(n) + O(1)$
  - But, by definition of $x_n$, $K(x_n) \geq n$ for each $n$, which is a contradiction for large enough $n$

# From Computation to Proofs

- These seemingly elementary considerations about Kolmogorov complexity point to deep issues in the foundations of mathematics!
- Recall Gödel's First Incompleteness Theorem: No consistent effectively axiomatizable proof system can prove all truths about the arithmetic of natural numbers
- We can get strong incompleteness results by arguing about Kolmogorov complexity in a similar way to how we showed uncomputability

# The Deep Intractability of Kolmogorov Complexity

- Theorem [C74]: Let $X$ be any effectively axiomatizable sound proof system. There are only finitely many $m$ for which a statement of the form "$K(x) \geq m$" that can be proved in $X$!
- Proof: Suppose, for the sake of contradiction, that there are infinitely many $m$ for which some statement "$K(x) \geq m$" is provable in $X$. Given $m$, we can computably find an $x$ such that "$K(x) \geq m$" is provable in $X$ by enumerating potential proofs of such statements in parallel until we find an actual one. But this $x$ has $K(x) \leq \log(m) + O(1)$, and for large enough $m$, this contradicts $K(x) \geq m$ (which is implied by the soundness of $X$)

# $K^{poly}$

- Let $U$ be a fixed time-efficient universal Turing machine, and let $t$ be a fixed polynomial
- $K^t(x) = \min\{|p|: U(p, \varepsilon) = x \text{ in at most } t(|x|) \text{ steps}\}$
- We have that for each $x$, $K^t(x) \leq |x| + O(1)$, and for each $n$, there is a string $x$ of length $n$ such that $K^t(x) \geq n$
- Note that $K(x) \leq K^t(x)$ for each $x$

# Meta-Complexity of $K^{poly}$

- Let $t$ be a fixed polynomial
  - $MK^tP$: Input is a string $x$ together with a parameter $s$, question is whether $K^t(x) \leq s$
  - $K^t$: Given a string $x$, compute the $K^t$ complexity of $x$
- MINKT: Input is a string $x$ together with parameters $s$ and $t$ in unary, question is whether $K^t(x) \leq s$
- $MK^tP$ and MINKT are in NP, and $K^t$ can be computed in poly time with an NP oracle
- Open whether any of these problems are NP-hard, however all of them are hard if one-way functions exist [RR97, KC00], and SZK reduces to them all [AD17]

# KS

- Let $U$ be a fixed space-efficient universal Turing machine
- $KS(x) = \min\{|p| + s : U(p, \varepsilon) = x \text{ using space at most } s\}$
- We have that for each $x$, $KS(x) \leq |x| + \log(|x|)$, and for each $n$, there is a string $x$ of length $n$ such that $KS(x) \geq n$
- Note that $K(x) \leq KS(x)$ for each $x$

# Meta-Complexity of KS

- MKSP: Input is a string x together with a parameter s, question is whether $KS(x) \leq s$
- KS: Given a string x, compute the KS complexity of x
- Observation: MKSP and KS are in polynomial space (by doing a brute-force search for the optimal program computing x)
- Theorem [ABKvMR06]: MKSP and KS are complete for PSPACE under non-uniform poly-size non-adaptive reductions and probabilistic poly-time Turing reductions
  - Note that this hardness is insufficient to establish that MKSP not in LOGSPACE, and indeed this is still an open question

# Meta-Complexity as a Tool: Relevance to Circuit Complexity

- Every NO instance (F,s) of MCSP corresponds to a circuit lower bound, i.e., that the Boolean function F does not have circuits of size s, and conversely all circuit lower bounds are encoded into the problem
- In recent work [A01, HS17, GIIKKT19, CKLM19], almost all known circuit lower bounds for explicit functions have been recovered for MCSP using connections with pseudorandomness
- MCSP and other meta-complexity problems are also associated with a "hardness magnification" phenomenon
  - If MCSP[$N^{o(1)}$] does not have circuits of size $N^{1.01}$, then NP ≠ P [MMW19, OPS19]

# Meta-Complexity as a Tool: Relevance to Learning Theory

- Learning is closely related to solving the search versions of MCSP and other meta-complexity problems
- Learning model: The learner is given oracle access to a target function and outputs a “good” hypothesis (i.e., small circuit) for the target function if such a hypothesis exists
- Search version of MCSP: Given a truth table, output a small circuit for the truth table if one exists
- Intuitively, if there is an efficient learner, one can solve (approximately) the search version of MCSP, simply using the input truth table to answer oracle queries
- [CIKK16] show that an efficient decision procedure for MCSP yields an efficient learner

# Meta-Complexity as a Tool: Relevance to Crypto

- Pseudo-random generators are essential to encryption and other cryptographic tasks
  - A pseudo-random generator (PRG) maps short random “seeds” to longer “pseudo-random” strings that are *computationally indistinguishable* from random strings
  - The outputs of a PRG have low complexity (since they can be generated from a short seed) while purely random strings do not (by a counting argument)
  - The security of a PRG relies on not being able to distinguish low-complexity strings from high-complexity ones
- This shows that pseudo-randomness implies the hardness of meta-complexity, but in fact the converse is also true [LP20, ILO22]

# Meta-Complexity as a Tool: Relevance to Proof Complexity and Meta-Mathematics

- Razborov and Rudich [RR97] identified an important *natural proofs* barrier to circuit lower bounds against strong circuit classes
  - They showed that known explicit lower bounds against weak circuit classes satisfy a certain naturalness property in a formal sense
  - They also showed that under standard cryptographic assumptions, there are no natural proofs against strong circuit classes such as the class of polynomial-size Boolean circuits
- The existence of natural proofs turns out to be *equivalent* to efficient zero-error average-case algorithms for MCSP! Thus the (presumed) intractability of MCSP is crucial to meta-mathematical barriers to circuit lower bounds