Meta-Complexity: Intro and a Brief Survey

Rahul Santhanam

(University of Oxford)

Plan of the Talk

- Introduction
- Vignette 1: Learning
- Vignette 2: Cryptography
- Vignette 3: Complexity Lower Bounds
- Further Directions

Plan of the Talk

- *Introduction*
- Vignette 1: Learning
- Vignette 2: Cryptography
- Vignette 3: Complexity Lower Bounds
- Further Directions

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 and Complexity Lower Bounds

- Showing that $\mathrm{DTIME}(2^{O(n)})$ does not have Boolean circuits of size $s(n)$ is equivalent to efficiently constructing NO instances of $\mathrm{MCSP}[s(\log(N))]$ 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 $2^{O(n)} = \mathrm{poly}(N)$ (where $N = 2^n$)
  - In the other direction, if L in $\mathrm{DTIME}(2^{O(n)})$ does not have Boolean circuits of size $s(n)$, then we can efficiently generate the truth table of L_n in time $2^{O(n)} = \mathrm{poly}(N)$, and this truth table is a NO instance of $\mathrm{MCSP}[s(\log(N))]$

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

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

Kolmogorov Complexity

- Let U be a fixed universal Turing machine
- For any string x in Σ^*, $K(x)$ is min $\{|p| : U(p, \varepsilon) = x\}$
  - Given y in Σ^*, $K(x|y)$ is min $\{|p| : U(p,y) = x\}$
- Basic properties
  - For every x in Σ^*, $K(x) \leq |x| + O(1)$
  - For each integer n, there is x of length n such that $K(x) \geq n$

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 K 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

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) ≤ 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

Plan of the Talk

- Introduction
- *Vignette 1: Learning*
- Vignette 2: Cryptography
- Vignette 3: Complexity Lower Bounds
- Further Directions

Search to Decision Reductions

- Let L be a problem in NP
- The decision problem for L is to decide, given x, whether x in L
- The search problem for L is to find, given x in L, a *proof* or *witness* that x in L
- Classical result: SAT is decidable in polynomial time iff the search problem for SAT is solvable in polynomial time
  - Proof idea: Iteratively determine the witness bit by bit, using one oracle call to the decision problem for each bit of the witness

Search to Decision for MCSP?

- The idea of the search-to-decision reduction for SAT doesn't seem to work for MCSP
  - Unclear how to find a circuit for a given truth table bit by bit just by asking questions about MCSP
- Until recently, nothing was known about whether search reduces to decision for MCSP
- The search version of MCSP is closely related to *learning*

Learning and MCSP

- Learning model: The learner is given oracle access to a target Boolean function F and outputs a “good” hypothesis (i.e., small circuit) C approximating the target function if there is a good hypothesis consistent with F
- Search version of MCSP: Given a truth table of a Boolean function F, output a small circuit C 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

Learning from Solving MCSP Efficiently

- Theorem [CIKK16]: Let C be a "reasonable" circuit class. If C-MCSP[$2^{n^{\varepsilon}}$] can be solved in time poly(N) (on average over the uniform distribution), then C-circuits of poly(n) size can be learned in time $2^{\text{polylog}(n)}$
- Corollary [CIKK16]: The class AC^0[Parity] of constant-depth unbounded fan-in circuits with Parity gates can be learned in quasi-polynomial time
  - Average-case algorithms for AC^0[Parity]-MCSP had been known since [RR97], based on lower bound techniques against AC^0[Parity]

Speedup for Learning

- Theorem [OS17]: Let C be a "reasonable" circuit class. There is $\varepsilon > 0$ such that C-circuits of $2^{n^{\wedge}\varepsilon}$ size can be learned in time $2^{O(n)}$ if and only if C-circuits of poly(n) size can be learned in time $2^{\text{polylog}(n)}$
- The statement of this result doesn't directly involve MCSP or meta-complexity, but the proof crucially uses the main result of [CIKK16]

Plan of the Talk

- Introduction
- Vignette 1: Learning
- *Vignette 2: Cryptography*
- Vignette 3: Complexity Lower Bounds
- Further Directions

One-Way Functions (OWFs)

- Efficient computability: f can be computed in polynomial time
- No efficient invertibility: There is no probabilistic poly-time procedure A that for most x, produces an inverse to f(x)

OWFs and Cryptography

- OWFs are the most fundamental primitive in theoretical cryptography
  - Cryptographic tasks such as private-key encryption, pseudorandom generation, bit commitment, message authentication and digital signatures are all *equivalent* to the existence of OWFs
- OWFs are based on various well-studied complexity assumptions such as the hardness of the Discrete Logarithm problem, Factoring problem and the Shortest Vector problem in certain lattices

Should We Believe in the Existence of OWFs?

- The existence of OWFs implies that NP ≠ P (and even the hardness of NP problems on average) but the reverse implication is unknown
- Problems such as Discrete Logarithm and Factoring are known to be efficiently solvable by quantum algorithms
- Other standard assumptions such as hardness of lattice problems could be much stronger than what we require

Characterizing OWFs using Meta-Complexity

- Liu and Pass [LP20] showed how to characterize OWFs using a natural average-case meta-complexity assumption
- Given a polynomial time bound t, we say that K^t is mildly hard on average over the uniform distribution if there is a polynomial p such that any probabilistic poly-time algorithm must fail to compute K^t on at least a $1/p(n)$ fraction of strings for large enough n
- Theorem [LP20]: Fix any polynomially bounded $t > 1.1\,n$. OWFs exist iff K^t is mildly hard on average over the uniform dist
- This is the first characterization of OWFs using average-case hardness of a natural problem

A Further Characterization of OWFs

- Theorem [IRS22]: The following are equivalent:
  - One-way functions exist
  - Kolmogorov complexity is hard to approximate on average over some "samplable" distribution, i.e., distribution sampled by some poly-time procedure
- Characterization based on hardness over *any* samplable distribution, while previous characterizations relied on the uniform distribution
- Works even for the uncomputable problem K!

Plan of the Talk

- Introduction
- Vignette 1: Learning
- Vignette 2: Cryptography
- *Vignette 3: Complexity Lower Bounds*
- Further Directions

Uniform vs Non-Uniform Lower Bounds

- Major open questions in complexity theory, such as the NP vs P question and the PSPACE vs P question, are about *uniform* lower bounds
- Since the 1980s, approaches to these questions have focused on showing stronger *non-uniform* lower bounds, i.e., that there is a problem in NP or in PSPACE that does not have polynomial-size Boolean circuits
  - These approaches have been largely unsuccessful and barriers such as the natural proof barrier [RR97] are known
- We are interested in new ways of exploiting the uniformity condition when proving lower bounds

Algorithmic Approaches to Lower Bounds

- While the area of complexity lower bounds has seen infrequent progress, research in algorithms is thriving [CKLPPS22, BNW22]
- Lower bounds are *impossibility* results while algorithms results are *possibility* results
- Counter-intuitive idea: Could we approach a lower bound by designing and analysing an algorithm for some computational task that we believe to be feasible?

Algorithmic Approaches to Lower Bounds

- Williams [W10] proposed an algorithmic approach to proving circuit lower bounds for NEXP (non-deterministic exponential time), and applied the approach [W11] to show that a new circuit lower bound for NEXP against ACC^0 circuits
- He showed in general that if SAT can be solved on C-circuits of size m on n variables in time $poly(m)2^{n-\omega(\log(n))}$, then NEXP does not have polynomial-size C-circuits

Algorithmic Approaches to Lower Bounds

- Williams' approach only has the potential to yield lower bounds against size s circuits for problems that require time more than s to solve, eg., lower bounds for exponential time against polynomial size
- However, in order to attack the NP vs P problem, we need to find an approach that applies to a problem solvable non-deterministically in some fixed polynomial amount of time (such as SAT) and yields arbitrary polynomial size lower bounds
- We give such an algorithmic approach, but for *uniform* rather than *non-uniform* lower bounds for PSPACE and NP

A Circuit-Based Sampling Task

- Input: A circuit C on n variables and of size $s = poly(n)$, such that C accepts at least a $2/3$ fraction of all inputs
- Task: Output some element of $SAT(C)$ with probability $>> 2^{-n}$
  - Here $SAT(C)$ is the set of satisfying assignments of C

- The trivial algorithm that outputs a random bitstring of length n runs in time n and outputs each element of $SAT(C)$ with probability 2^{-n}
  - Can we find an algorithm that is almost as efficient but beats random guessing for some element of $SAT(C)$?

A Simulation-Based Algorithm

- Input: A circuit C on n variables and of size $s = poly(n)$, such that C accepts at least a $2/3$ fraction of all inputs
- Task: Output some element of $SAT(C)$ with probability $>> 2^{-n}$
  - Here $SAT(C)$ is the set of satisfying assignments of C

- The following simple algorithm runs in time (and space) $O(sn^5)$ and outputs some element of $SAT(C)$ with probability $>= n^4/2^n$: pick n^5 strings of length n independently and uniformly at random, and output the lexicographically first one that satisfies C

An Algorithmic Approach

Input: A circuit C on n variables of size $\text{poly}(n)$, accepting $\geq 2/3$ fraction of inputs

Task: Output some fixed satisfying input y of C with probability $\geq n^4/2^n$, using space $O(n^2)$

Theorem [S23]: If the task is solvable, then PSPACE ≠ P

- This gives an *algorithmic* formulation of the PSPACE ≠ P problem, which is about *lower bounds*
- Proof of the implication uses meta-complexity

An Algorithmic Approach

Input: A circuit C on n variables of size $\text{poly}(n)$, accepting $\geq 2/3$ fraction of inputs, described by a *compressed* representation of size n

Task: Output some fixed satisfying input y of C with probability $\geq n^4/2^n$, using time $O(n^2)$

Theorem [S23]: If the task is solvable, then $\text{NP} \neq \text{P}$

- This gives an *algorithmic* formulation of the $\text{NP} \neq \text{P}$ problem, which is about *lower bounds*
- Proof of the implication uses meta-complexity

Features of the Approach

- It is an approach to NP vs P that exploits the power of NP
  - Several previous approaches to circuit lower bounds for circuit classes C yielded hard functions in P against C, and therefore are not useful in the most general setting
- It exploits uniformity of the lower bound
  - Previous approaches applied to non-uniform lower bounds and ran up against the natural proofs barrier [RR97]
  - It is possible that uniform lower bounds are much easier to prove than non-uniform ones
- It is very general, applying to any circuit class C, and therefore could be useful in making gradual progress

Proof Template

- Reminder of circuit-based sampling task for PSPACE lower bounds
  - Given: A circuit C on n variables of size $\text{poly}(n)$, accepting $\geq 2/3$ fraction of inputs
  - Output: Some fixed satisfying input y of C with probability $\geq n^4/2^n$
  - The algorithm should use space $O(n^2)$
- Theorem: If the circuit-based sampling task is solvable, then PSPACE $\neq$ P
- The statement of the theorem does not involve meta-complexity, but the proof will use meta-complexity as a tool

Proof Template

- Theorem: If the circuit-based sampling task is solvable, then PSPACE ≠ P
- We assume, for the sake of contradiction, that PSPACE = P
- We consider a version of Kolmogorov complexity called *probabilistic time-bounded Kolmogorov complexity* pK^{poly} [GKLO22, LOZ22]
  - Informally, the pK^{poly} complexity of a string x is the size of the smallest program that can generate x in polynomial time given access to a random string
- Let R be the set of strings with pK^{poly} complexity at least n-1
- Easy to show that R includes at least half the strings of length n

Proof Template

- Theorem: If the circuit-based sampling task is solvable, then PSPACE ≠ P
- Let R be the set of strings with pK^{poly} complexity at least n-5
- Easy to show that R includes at least half the strings of length n and also that R is in PSPACE
- Since PSPACE = P, we have that R has uniform Boolean circuits $\{C_n\}$, where $pK^{poly}(C_n)$ is at most log(n) + O(1) by uniformity
- By the solvability of the circuit sampling task, we can show that there is a string y accepted by C_n such that $pK^{poly}(y|C_n)$ is at most n-3log(n)
- Therefore $pK^{poly}(y)$ is at most n-log(n) for large n, which contradicts the assumption that y ε R

Necessity of the Approach

- Theorem: Under standard circuit lower bound assumptions for exponential time (i.e., that DTIME($2^{O(n)}$) requires circuits of size $2^{\Omega(n)}$), PSPACE ≠ P if and only if the sampling task is solvable
- Thus the approach is without loss of generality if we believe in strong circuit lower bounds

Applications of the Approach

- The approach can be used to give new proofs of old results such as the space hierarchy theorem and Allender's uniform lower bound for the Permanent [A99]
- It can also be used to show some new uniform lower bounds in NP (but still very far off from saying anything interesting about NP vs P)

Plan of the Talk

- Introduction
- Vignette 1: Learning
- Vignette 2: Cryptography
- Vignette 3: Complexity Lower Bounds
- *Further Directions*

Some Open Problems

- Characterize precisely the complexity status of problems such as MCSP and MK^tP
- Under standard complexity assumptions, can we generate K^{poly}-random strings in polynomial time?
  - This is relevant to the question of *explicit constructions* of combinatorial and number-theoretic objects such as Ramsey graphs, error-correcting codes
- The Allender program: characterize complexity classes in terms of resource-bounded reductions to the Kolmogorov-random strings
  - For example, NEXP in BPP^K in EXPSPACE
- Show unprovability results for statements of the form "x is a K^{poly}-random string", i.e., resource-bounded versions of Chaitin's theorem