Meta Complexity

Lecture 1

Ronald de Haan
me@ronalddehaan.eu

University of Amsterdam

January 8, 2023

The project organization in a nutshell

- Introductory lectures and Q&A sessions
- Recorded video lectures from the Simons Institute
- You study a paper (possibly in pairs), and present it to the group
- You write a final report

What will we cover in this lecture?

- What is meta complexity?
- Basic observations and results about MCSP
- A brief primer on Kolmogorov complexity

What is meta complexity?

- Meta complexity is an informal term referring to the computational complexity study of problems that have a ‘complexity flavor’
- So in a sense, meta complexity studies the complexity of complexity problems (hence the phrase ‘meta’)
- This turns out to be fruitful for studying various notions related to computational complexity, learning, cryptography, etc.

The Minimum Circuit Size Problem (MCSP)

- MCSP:
  - *Input:* a Boolean function F over n variables given by its truth table (containing 2^n entries), and a positive integer $s \in \mathbb{N}$ (given in binary).
  - *Question:* does there exist a Boolean circuit C of size s that expresses the function F?

- MCSP[s], for a function $s : \mathbb{N} \to \mathbb{N}$:
  - *Input:* a Boolean function F over n variables given by its truth table (containing 2^n entries).
  - *Question:* does there exist a Boolean circuit C of size $s(2^n)$ that expresses the function F?

Black-box problem

- Intuitively, MCSP is a black-box problem:
  - We are given the input-output behavior of a function F
  - The task is to see if this function F has small circuits

- Compare this to white-box problems such as SAT, where we are given an explicit way to compute the Boolean function F about which we are answering a question—namely, by means of a formula or circuit

MCSP is in NP

- MCSP is in NP
- Might seem odd at first:
  - Circuits to consider are exponentially large in the size of (the binary encoding of) s
- Main idea:
  - There is always a circuit for F of size $O(2^n)$
  - We are given the truth table of F as input, which is of size 2^n
  - So we can guess a circuit C of size at most $O(2^n)$ in polynomial time
  - And check if C expresses F by iterating over all rows α in the truth table, and checking if $C(\alpha) = F(\alpha)$

Open question: is MCSP in P? Is it NP-complete?

- One main open research question:

Is MCSP **NP-complete?**

- MCSP is not in P assuming one-way functions exist.
  - *More on this later..*

A connection between MCSP and circuit lower bounds

- The following two are equivalent:
  - Showing that $\mathsf{DTIME}(2^{O(n)})$ does not have Boolean circuits of size $s(n)$
  - Efficiently (in polynomial time) constructing no-instances of $\mathsf{MCSP}[s']$—where $s' = s \circ \log$—of size 2^n,given 2^n in unary.
- Main idea:
  - Suppose there is a problem L in $\mathsf{DTIME}(2^{O(n)})$ that has no circuits of size $s(n)$.
    Using this, we can compute in time $2^{O(n)} = \mathrm{poly}(2^n)$ the truth table of problem L on inputs of size n.
    This is a no-instance of $\mathsf{MCSP}[s']$ of size 2^n.
  - Suppose you can efficiently construct no-instances of $\mathsf{MCSP}[s']$ of size 2^n.
    Using this, for each input size, we can construct (in exponential time) a truth table of a Boolean function that has no circuits of size $s(n)$.
    This yields a problem in $\mathsf{DTIME}(2^{O(n)})$ that has no circuits of size $s(n)$.

Kolmogorov complexity: origins in randomness

- One of the main roots of Kolmogorov complexity is the study of randomness
- Consider the strings 000000000000 and 011011110010, both of length 12.
  - Is one more ‘random’ than the other?
- How do we measure this? Perhaps considering a probability distribution over all strings of length 12 and considering the probability of the strings. The uniform distribution doesn’t help to define randomness.
- Idea of Kolmogorov complexity: measure the amount to which strings can be compressed.

Kolmogorov complexity

- Pick some universal Turing machine $\mathbb{U}$.
- The Kolmogorov complexity $C(x)$ of a string x is defined as:

$$C(x) = \min\{\ |p| : \mathbb{U}(p) = x\ \}.$$

- In other words, the Kolmogorov complexity $C(x)$ of x is the size of the smallest program p that, when executed by $\mathbb{U}$, yields x as output.

Invariance Theorem

- The definition of the Kolmogorov complexity C depends on the choice of the UTM $\mathbb{U}$. But:
- The **Invariance Theorem** states that for any two UTMs $\mathbb{U}_1, \mathbb{U}_2$
there is some constant $c \in \mathbb{N}$ (depending only on $\mathbb{U}_1, \mathbb{U}_2$)
such that for all strings x it holds that $C_{\mathbb{U}_2}(x) \leq C_{\mathbb{U}_1}(x) + c$.
  - Main idea: give $\mathbb{U}_2$ a description of $\mathbb{U}_1$ and a program p for $\mathbb{U}_1$, together with instructions to simulate $\mathbb{U}_1$ on p.
- In other words, up to some additive constant, the choice of which UTM to use does not matter.

A basic upper bound on the Kolmogorov complexity of strings

- For each string $x \in \{0,1\}^n$ it holds that $C(x)$ is $O(n)$.
  - Main idea: construct a program p that contains x explicitly and the instruction to print x.
  - The size of this program is n (to write down x) plus some additional constant (for the instructions to print out x).

Strings with high Kolmogorov complexity exist

- For each $n \in \mathbb{N}$, there exists a string $x \in \{0,1\}^n$ such that $C(x) \geq n$.
  - Main idea: counting.
  - There are 2^n strings $x \in \{0,1\}^n$.
  - The number of programs p of length $< n$ is $\sum_{i=0}^{n-1} 2^i = 2^n - 1$.
  - By the pigeonhole principle, there must be at least one string $x \in \{0,1\}^n$ with $C(x) \geq n$.

Kolmogorov complexity is uncomputable

- The problem of computing the Kolmogorov complexity $C(x)$ of a string x is uncomputable.
  - Main idea: an incompressibility argument.
  - Suppose, to derive a contradiction, that C is computable.
  - Consider the following algorithm $\mathbb{A}_M$, whose description will be of length $P + \log M$:
    - Iterate over all strings $x \in \{0,1\}^*$, from shortest to longer.
    - For each string x, compute $C(x)$. If $C(x) \geq M$, return x.
    - (In other words, $\mathbb{A}_M$ returns the first string x with $C(x) \geq M$.)
  - Now select M such that $M > P + \log M$.
  - Let x be the string that $\mathbb{A}_M$ returns. So $C(x) \leq P + \log M < M$. This contradicts that $C(x) \geq M$.

Time-bounded Kolmogorov complexity

- Resource-bounded variants of Kolmogorov complexity have been considered.
- Let $t : \mathbb{N} \to \mathbb{N}$.
- Then:

$$C^t(x) = \min\{\ |p| : \mathbb{U}(p) = x \text{ in time } t(|x|)\ \}.$$

- Observation: for each x and each t, it holds that $C(x) \leq C^t(x)$.

Levin's Kt complexity

- Levin's Kt complexity is another variant that is based on time bounds.
- It is defined as follows:

$$Kt(x) = \min\{\ |p| + \log t : \mathbb{U}(p) = x \text{ in time } t\ \}.$$

- Observation: for each x, it holds that $C(x) \leq Kt(x)$.

Computational problems: MINKT, MK^{t}P, and MKtP

- MINKT: given a string x and $s, t \in \mathbb{N}$ in unary, decide whether there is a program p of size $\leq s$ such that $\mathbb{U}(p) = x$ in time t.
  - in NP
- MK^{t}P: given a string x and $s \in \mathbb{N}$ in unary, decide whether there is a program p of size $\leq s$ such that $\mathbb{U}(p) = x$ in time $t(|x|)$.
  - in NP
- MKtP: given a string x and $s \in \mathbb{N}$ in unary, decide whether $Kt(x) \leq s$.
  - in EXP

A note on notation

- There is also a variant of Kolmogorov complexity called prefix complexity, that is based on prefix-free codes
  - (This has some theoretical advantages over the classical definition, in some settings)
- Typically, the letter K is used to denote (variants of) prefix complexity, and the letter C is used for the classical versions—but this differs from text to text.
- More generally, notations may differ slightly from one text to the other, so be aware. :-)

Summary

- What is meta complexity?
- Basic observations and results about MCSP
- A brief primer on Kolmogorov complexity