Randomness and Intractability in Kolmogorov Complexity

Igor Carboni Oliveira

University of Oxford

ICALP 2019

Background and motivation

Structure versus Randomness

- Given a string $x \in \{0,1\}^n$, is it "**structured**" or "**random**"?

- Question of relevance to several fields, including:

LEARNING:	Detecting pattern/structure in data.
CRYPTO:	Encrypted strings must look random.

Complexity of strings

- ▷ Different ways of measuring the complexity of x.

0100110001110000111100000111110000001111110000000111111100000000011111111

```c
#include <stdio.h>
int main()
{
    int i,j;
    for( i = 1; i <= 8; i = i + 1 )
    {
        for (j=1; j<=i; j = j + 1){ printf("0"); }
        for (j=1; j<=i; j = j + 1){ printf("1"); }
    }
    return 0;
}
```

- ▷ **This talk:** Interested in **hardness** of estimating complexity.

- ▷ Different ways of measuring the complexity of x.

010011000111000011110000011111000000111111000000011111110000000011111111

```c
#include <stdio.h>
int main()
{
    int i,j;
    for( i = 1; i <= 8; i = i + 1 )
    {
        for (j=1; j<=i; j = j + 1){ printf("0"); }
        for (j=1; j<=i; j = j + 1){ printf("1"); }
    }
    return 0;
}
```

- ▷ **This talk:** Interested in **hardness** of estimating complexity.

If provably secure cryptography exists, algorithms shouldn't be able to estimate the "complexity" of strings.

Circuit Complexity:

- View x as a boolean function $f\colon \{0,1\}^\ell \to \{0,1\}$.
- $\mathsf{complexity}(x)$ = minimum size of a circuit for f.
- Deciding complexity is just the MCSP. Showing this is hard implies **P** $\neq$ **NP**.

Circuit complexity and Kolmogorov complexity

Circuit Complexity:

- View x as a boolean function $f\colon \{0,1\}^\ell \to \{0,1\}$.
- $\mathsf{complexity}(x)$ = minimum size of a circuit for f.
- Deciding complexity is just the MCSP. Showing this is hard implies **P** $\neq$ **NP**.

Kolmogorov Complexity:

- $\mathsf{complexity}(x)$ = minimum length of TM that prints x.
- Estimating complexity of x is **undecidable**.

Circuit Complexity:

- View x as a boolean function $f\colon \{0,1\}^\ell \to \{0,1\}$.
- complexity(x) = minimum size of a circuit for f.
- Deciding complexity is just the MCSP. Showing this is hard implies **P** $\neq$ **NP**.

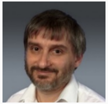
Kolmogorov Complexity:

- complexity(x) = minimum length of TM that prints x.
- Estimating complexity of x is **undecidable**.

"Extremal" . . . Is there an intermediate notion that is useful?

Time-bounded Kolmogorov complexity

- Introduced by L. Levin in 1984.

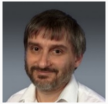


- Takes into account **description length** and **running time** of TM.

$$\mathrm{Kt}(x) \stackrel{\text{def}}{=} \min_{\substack{\text{TM } M, \text{ time } t \\ M \text{ prints } x \text{ in time } t}} |M| + \log t$$

Time-bounded Kolmogorov complexity

- ▷ Introduced by L. Levin in 1984.



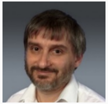
- ▷ Takes into account **description length** and **running time** of TM.

$$\mathsf{Kt}(x) \overset{\text{def}}{=} \min_{\substack{\text{TM } M, \text{ time } t \\ M \text{ prints } x \text{ in time } t}} |M| + \log t$$

- ▷ $\mathsf{Kt}(x)$ can be computed in exponential time (brute-force).

Time-bounded Kolmogorov complexity

- ▷ Introduced by L. Levin in 1984.



- ▷ Takes into account **description length** and **running time** of TM.

$$\mathsf{Kt}(x) \stackrel{\text{def}}{=} \min_{\substack{\text{TM } M,\text{ time } t \\ M \text{ prints } x \text{ in time } t}} |M| + \log t$$

- ▷ Kt(x) can be computed in exponential time (brute-force).

Circuit Complexity	**Levin's (Time-Bounded) Kt**	Kolmogorov Complexity
NP	EXP	undecidable

Why is Kt an interesting measure?

- $\log t$ gives the "right" measure: connection to **optimal search**.

Example: Deterministic generation of n-bit prime numbers.

Fastest known algorithm runs in time $2^{n/2}$ [Lagarias-Odlyzko, 1987].

Why is Kt an interesting measure?

- ▷ $\log t$ gives the "right" measure: connection to **optimal search**.

Example: Deterministic generation of n-bit prime numbers.

Fastest known algorithm runs in time $2^{n/2}$ [Lagarias-Odlyzko, 1987].

- ▷ Is there a sequence $\{p_n\}$ of n-bit primes such that $\mathsf{Kt}(p_n) = o(n)$?

Why is Kt an interesting measure?

- $\log t$ gives the "right" measure: connection to **optimal search**.

Example: Deterministic generation of n-bit prime numbers.

Fastest known algorithm runs in time $2^{n/2}$ [Lagarias-Odlyzko, 1987].

- Is there a sequence $\{p_n\}$ of n-bit primes such that $\mathsf{Kt}(p_n) = o(n)$?

True $\iff$ there is deterministic prime generation in time $2^{o(n)}$

How difficult is to compute the complexity of a string?

Can we compute $\mathsf{Kt}(x)$ in polynomial time?

- ▷ Explicitly posed in [ABK+06]. We already know that P $\neq$ EXP . . .
- ▷ Question strongly connected to power of learning algorithms.
- ▷ If provably secure cryptography exists, the answer should be **negative**.

Main Result

Summary of Main Contribution

- ▷ We introduce a **randomized analogue** of Levin's Kt complexity.

- ▷ **Main Result:** Randomized Kt complexity **cannot** be estimated in BPP.

(The problem can be solved in randomized exponential time.)

- ▷ This is an **unconditional** lower bound for a natural problem.

Randomized Kt Complexity

- ▷ Adaptation of Levin's definition to **Randomized Computation**.

- ▷ For $x \in \{0,1\}^n$, we consider algorithms that generate x w.h.p.:

$$\mathsf{rKt}(x) \stackrel{\text{def}}{=} \min_{\substack{\underline{\text{randomized}}\text{ TM M, time t} \\ \Pr_M[\, M \text{ prints } x \text{ in time } t\,] \geq 2/3}} |M| + \log t$$

Intuition: String probabilistically decompressed from short representation.

Remarks about Kt Complexity

$$\text{rKt}(x) \stackrel{\text{def}}{=} \min_{\substack{\underline{\text{randomized}}\ \text{TM M, time t} \\ \Pr_M[\,M \text{ prints } x \text{ in time } t\,] \geq 2/3}} |M| + \log t$$

- ▷ Definition is **robust**.

Remarks about Kt Complexity

$$\text{rKt}(x) \stackrel{\text{def}}{=} \min_{\substack{\text{\underline{randomized} TM M, time t} \\ \Pr_M[\,M \text{ prints } x \text{ in time } t\,] \geq 2/3}} |M| + \log t$$

▷ Definition is **robust**.

▷ Connected to **pseudodeterministic algorithms**.
In particular, it follows from a recent joint work with R. Santhanam that

– There is an infinite sequence $\{p_m\}_m$ of m-bit primes such that $\text{rKt}(p_m) \leq m^{o(1)}$.

Remarks about Kt Complexity

$$\text{rKt}(x) \stackrel{\text{def}}{=} \min_{\substack{\underline{\text{randomized}}\text{ TM M, time t} \\ \Pr_M[\,M \text{ prints } x \text{ in time } t\,] \geq 2/3}} |M| + \log t$$

▷ Definition is **robust**.

▷ Connected to **pseudodeterministic algorithms**.
In particular, it follows from a recent joint work with R. Santhanam that

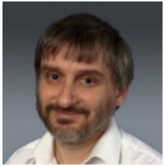
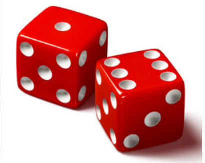
– There is an infinite sequence $\{p_m\}_m$ of m-bit primes such that $\text{rKt}(p_m) \leq m^{o(1)}$.

▷ Under standard derandomization assumptions, $\textbf{Kt}(x) = \Theta(\textbf{rKt}(x))$.

How difficult is to compute the complexity of a string?

Can we compute $\mathsf{Kt}(x)$ **in polynomial time?**

MKtP – Minimum Kt Problem

Can we compute $\mathsf{rKt}(x)$ **in randomized polynomial time?**

MrKtP – Minimum rKt Problem

Main Result: MrKtP is hard

“rKt cannot be approximated in quasi-polynomial time.”

Theorem 1. *For every $\varepsilon > 0$, there is no randomized algorithm running in time $n^{\mathsf{poly}(\log n)}$ that distinguishes between $\mathsf{rKt}(x) \leq n^{\varepsilon}$ versus $\mathsf{rKt}(x) \geq .99n$, where n is the length of the input string x.*

Remark. This problem **can** be solved in randomized **exponential** time.

Techniques

Preliminaries

Gap-MrKtP$[n^{\varepsilon}, .99n]$:

$$\mathcal{YES}_n \stackrel{\text{def}}{=} \{x \in \{0,1\}^n \mid \mathsf{rKt}(x) \leq n^{\varepsilon}\}$$

$$\mathcal{NO}_n \stackrel{\text{def}}{=} \{x \in \{0,1\}^n \mid \mathsf{rKt}(x) > .99n\}$$

▷ Algorithm for Gap-MrKtP$[n^{\varepsilon}, .99n]$ distinguishes two cases.

Gap-MrKtP$[n^\varepsilon, .99n]$:

$$\mathcal{YES}_n \stackrel{\text{def}}{=} \{x \in \{0,1\}^n \mid \mathsf{rKt}(x) \leq n^\varepsilon\}$$

$$\mathcal{NO}_n \stackrel{\text{def}}{=} \{x \in \{0,1\}^n \mid \mathsf{rKt}(x) > .99n\}$$

$\triangleright$ Algorithm for Gap-MrKtP$[n^\varepsilon, .99n]$ distinguishes two cases.

$\triangleright$ Approach: **indirect diagonalization** using techniques from **theory of pseudorandomness**.

Lemma 1. For every $\varepsilon > 0$, $\mathsf{BPE} \leq_{\mathsf{P/poly}} \mathsf{Gap\text{-}MrKtP}[n^{\varepsilon}, .99n]$.

▷ Very strong **non-uniform inclusion**.

Main Lemmas

Lemma 1. For every $\varepsilon > 0$, $\mathsf{BPE} \leq_{\mathsf{P/poly}} \mathsf{Gap\text{-}MrKtP}[n^{\varepsilon}, .99n]$.

▷ Very strong **non-uniform inclusion.**

Lemma 2. For every $\varepsilon > 0$, $\mathsf{PSPACE} \subseteq \mathsf{BPP}^{\mathsf{Gap\text{-}MrKtP}[n^{\varepsilon}, .99n]}$.

▷ Strong **uniform inclusion.**

Lemma 1. For every $\varepsilon > 0$, $\mathsf{BPE} \leq_{\mathsf{P/poly}} \mathsf{Gap\text{-}MrKtP}[n^{\varepsilon}, .99n]$.

▷ Very strong **non-uniform inclusion**.

Lemma 2. For every $\varepsilon > 0$, $\mathsf{PSPACE} \subseteq \mathsf{BPP}^{\mathsf{Gap\text{-}MrKtP}[n^{\varepsilon}, .99n]}$.

▷ Strong **uniform inclusion**.

Lemma 3. If $n \leq s(n) \leq 2^{o(n)}$ then $\mathsf{DSPACE}[s^3] \not\subseteq \mathsf{Circuit}[s]$.

▷ Nexus between **uniform** and **non-uniform** inclusions.

Main Result from Lemmas 1, 2, and 3

▷ **Proof by contradiction.** Sketch of weaker result:

Assume $\mathsf{Gap\text{-}MrKtP}[n^{\varepsilon}, .99n] \in \mathsf{BPP}$. This also gives inclusion in $\mathsf{P/poly}$.

L1. $\mathsf{BPE} \leq_{\mathsf{P/poly}} \mathsf{Gap\text{-}MrKtP}[n^{\varepsilon}, .99n]$. This implies $\mathsf{BPE} \subseteq \mathsf{Circuit}[\mathsf{poly}]$.

L2. $\mathsf{PSPACE} \subseteq \mathsf{BPP}^{\mathsf{Gap\text{-}MrKtP}[n^{\varepsilon}, .99n]}$. This implies $\mathsf{PSPACE} \subseteq \mathsf{BPP}$.

Translation gives $\mathsf{DSPACE}[n^{\mathsf{poly}(\log n)}] \subseteq \mathsf{BPTIME}[n^{\mathsf{poly}(\log n)}] \subseteq \mathsf{BPE} \subseteq \mathsf{Circuit}[\mathsf{poly}]$.

This inclusion contradicts **L3.** $\mathsf{DSPACE}[s^3] \not\subseteq \mathsf{Circuit}[s]$. □

Theory of Pseudorandomness – Intuition for Lemmas 1 and 2

▷ **Hardness versus Randomness paradigm:**

From “hard” $f\colon \{0,1\}^m \to \{0,1\}$, one designs a “pseudorandom generator”

$$G^f\colon \{0,1\}^\ell \to \{0,1\}^n.$$

Theory of Pseudorandomness – Intuition for Lemmas 1 and 2

▷ **Hardness versus Randomness paradigm:**

From "hard" $f\colon \{0,1\}^m \to \{0,1\}$, one designs a "pseudorandom generator"

$$G^f\colon \{0,1\}^\ell \to \{0,1\}^n.$$

Proof often shows: Algorithm "breaking" G^f can be used to "compute" f.

▷ **Hardness versus Randomness paradigm:**

From "hard" $f\colon \{0,1\}^m \to \{0,1\}$, one designs a "pseudorandom generator"

$$G^f\colon \{0,1\}^\ell \to \{0,1\}^n.$$

Proof often shows: Algorithm "breaking" G^f can be used to "compute" f.

Crucial: We can upper bound rKt complexity of output strings of G^f.

Algorithm solving Gap-MrKtP$[n^\varepsilon, .99n]$ acts as a **distinguisher**!

Theory of Pseudorandomness – Intuition for Lemmas 1 and 2

L1. $\mathsf{BPE} \leq_{\mathsf{P/poly}} \mathsf{Gap\text{-}MrKtP}[n^{\varepsilon}, .99n]$. Relies on PRG construction of **[BFNW93]**.

L2. $\mathsf{PSPACE} \subseteq \mathsf{BPP}^{\mathsf{Gap\text{-}MrKtP}[n^{\varepsilon}, .99n]}$. Relies on PRG construction of **[TV07]**.

Theory of Pseudorandomness – Intuition for Lemmas 1 and 2

L1. BPE $\leq_{\mathsf{P/poly}}$ Gap-MrKtP$[n^{\varepsilon}, .99n]$. Relies on PRG construction of **[BFNW93]**.

L2. PSPACE $\subseteq$ BPP$^{\text{Gap-MrKtP}[n^{\varepsilon},.99n]}$. Relies on PRG construction of **[TV07]**.

- ▷ **L1** and variants: require notions of string complexity such as **rKt** and **Kt**.
- ▷ **Randomness is used** in the proof of **L2**: bottleneck to Levin's **Kt**.

Further Results

(**uniform** versus **non-uniform** lower bounds)

Circuit lower bounds

- ▷ Lower bound presented before holds against **uniform** algorithms.
- ▷ Boolean circuits capture **non-uniform** computation.

Major Challenge: Show for an explicit problem that any circuit solving the problem requires several AND, OR, NOT gates.

State-of-the-art circuit lower bounds

After 50+ years of intensive investigation:

- $\triangleright$ Existing circuit lower bounds are of the form $c \cdot n$ for constant c.
- $\triangleright$ Boolean formulas (weaker model): lower bounds of the form $n^{3-o(1)}$.

State-of-the-art circuit lower bounds

After 50+ years of intensive investigation:

▷ Existing circuit lower bounds are of the form $c \cdot n$ for constant c.

▷ Boolean formulas (weaker model): lower bounds of the form $n^{3-o(1)}$.

We know that $\mathsf{Gap\text{-}MrKtP}[n^{\varepsilon}, .99n]$ is hard. Can we use it to prove better circuit and formula lower bounds?

Hardness Magnification

▷ Emerging theory showing that **weak** lower bounds can be “magnified” to **strong** lower bounds.

Hardness Magnification

▷ Emerging theory showing that **weak** lower bounds can be “magnified” to **strong** lower bounds.

▷ By adapting recent joint work with J. Pich and R. Santhanam:

Theorem 2. *If for every* $\varepsilon > 0$*,*

$\textsf{Gap-MrKtP}[n^{\varepsilon}, .99n] \notin \textsf{Circuit}[n^{1.01}]$, then BPEXP $\not\subseteq \textsf{Circuit}[\textsf{poly}]$.

$\textsf{Gap-MrKtP}[n^{\varepsilon}, .99n] \notin \textsf{Formula}[n^{3.01}]$, then BPEXP $\not\subseteq \textsf{Formula}[\textsf{poly}]$.

Open Problems

The deterministic case

▷ Can we prove that computing Levin's Kt complexity cannot be done in deterministic polynomial time?

Power of randomness: NEXP versus BPP

▷ **This work:** natural problem that cannot be solved in randomized quasi-polynomial time.

▷ Can we reduce **approximating** rKt to a problem in **NEXP**?

▷ Even a randomized reduction would show **NEXP** $\neq$ **BPP**.

References and related work

- Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger.
  Power from random strings.
  SIAM J. Comput., 35(6):1467–1493, 2006.
- Eric Allender, Michal Koucký, Detlef Ronneburger, and Sambuddha Roy.
  The pervasive reach of resource-bounded Kolmogorov complexity in computational complexity theory.
  J. Comput. Syst. Sci., 77(1):14–40, 2011.
- Eric Allender.
  The complexity of complexity.
  In *Computability and Complexity*, pages 79–94. Springer, 2017.
- László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson.
  BPP has subexponential time simulations unless EXPTIME has publishable proofs.
  Computational Complexity, 3:307–318, 1993.
- Eran Gat and Shafi Goldwasser.
  Probabilistic search algorithms with unique answers and their cryptographic applications.
  Electronic Colloquium on Computational Complexity (ECCC), 18:136, 2011.
- Leonid A. Levin.
  Randomness conservation inequalities; information and independence in mathematical theories.
  Information and Control, 61(1):15–37, 1984.
- Ming Li and Paul M. B. Vitányi.
  An Introduction to Kolmogorov Complexity and Its Applications.
  Texts in Computer Science. Springer, 2008.
- Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam.
  Hardness magnification near state-of-the-art lower bounds.
  Computational Complexity Conference (CCC), 2019.
- Igor Carboni Oliveira and Rahul Santhanam.
  Pseudodeterministic constructions in subexponential time.
  In *Symposium on Theory of Computing (STOC)*, pages 665–677, 2017.
- Luca Trevisan and Salil P. Vadhan.
  Pseudorandomness and average-case complexity via uniform reductions.
  Computational Complexity, 16(4):331–364, 2007.