The Computational Complexity Column

by

Michal Koucký

Computer Science Institute, Charles University
Malostranské nám. 25, 118 00 Praha 1, Czech Republic
koucky@iuuk.mff.cuni.cz
https://iuuk.mff.cuni.cz/~koucky/

THEORY AND APPLICATIONS OF PROBABILISTIC KOLMOGOROV COMPLEXITY

Zhenjian Lu[*] Igor C. Oliveira[†]

Abstract

Diverse applications of Kolmogorov complexity to learning [CIKK16], circuit complexity [OPS19], cryptography [LP20], average-case complexity [Hir21], and proof search [Kra22] have been discovered in recent years. Since the running time of algorithms is a key resource in these fields, it is crucial in the corresponding arguments to consider *time-bounded* variants of Kolmogorov complexity. While fruitful interactions between time-bounded Kolmogorov complexity and different areas of theoretical computer science have been known for quite a while (e.g., [Sip83, Ko91, ABK$^+$06, AF09], to name a few), the aforementioned results have led to a renewed interest in this topic.

The theory of Kolmogorov complexity is well understood, but many useful results and properties of Kolmogorov complexity are not known to hold in time-bounded settings. Unfortunately, this creates technical difficulties or leads to conditional results when applying methods from time-bounded Kolmogorov complexity to algorithms and complexity theory. Perhaps even more importantly, in many cases it is desirable or even necessary to consider *randomised* algorithms. Since random strings have high complexity, the classical theory of time-bounded Kolmogorov complexity might be inappropriate or simply cannot be applied in such contexts.

To mitigate these issues and develop a more robust theory of time-bounded Kolmogorov complexity that survives in the important setting of randomised computations, some recent papers [Oli19, LO21, LOS21, GKLO22, LOZ22] have explored *probabilistic* notions of time-bounded Kolmogorov complexity, such as rKt complexity [Oli19], rK^t complexity [LOS21], and pK^t complexity [GKLO22]. These measures consider different ways of encoding an object via a *probabilistic representation*. In this survey, we provide an introduction to probabilistic time-bounded Kolmogorov complexity and its applications, highlighting many open problems and research directions.

[*]University of Warwick, UK. E-mail: zhen.j.lu@warwick.ac.uk

[†]University of Warwick, UK. E-mail: igor.oliveira@warwick.ac.uk

1 Introduction

Consider an arbitrary binary string $x \in \{0,1\}^*$, e.g.,

$$x = 1010101010101010101011110010110000101011\,. \qquad (1)$$

The Kolmogorov complexity of x, $\mathsf{K}(x)$, is the length $|M|$ of the shortest program M that prints x when computing over the empty input string.[1] Intuitively, $\mathsf{K}(x)$ can be seen as a measure of the "randomness" of x, in the sense that simple strings exhibiting an apparent pattern have bounded Kolmogorov complexity (e.g., the leftmost 20 bits of the string x from Equation (1)), while a typical random n-bit string has $\mathsf{K}(x)$ close to n, i.e., it cannot be compressed. The investigation of Kolmogorov complexity has uncovered surprising connections to distant areas of mathematics and computer science, ranging from computability, logic, and algorithm design to number theory, combinatorics, statistics and a number of other fields. We refer to [SUV17, LV19] for a comprehensive treatment of Kolmogorov complexity and its applications.

Despite the appealing nature and wide applicability of Kolmogorov complexity, its results and techniques tend to be inappropriate in settings where the *running time of algorithms is of concern*, e.g., in complexity theory, computational learning theory, and cryptography. This is because $\mathsf{K}(x)$ does not take into account the time that the machine M takes to output x. To address this issue, several authors have contributed to the development of *time-bounded* Kolmogorov complexity. In order to proceed with our discussion, we describe two prominent time-bounded Kolmogorov complexity notions. (A formal treatment appears in Section 2.)

In an influential paper, Levin [Lev84] introduced $\mathsf{Kt}(x)$, a variant of Kolmogorov complexity that simultaneously takes into account the running time t and description length $|M|$ of all programs M that output x. More precisely, given a string $x \in \{0,1\}^*$, we let

$$\mathsf{Kt}(x) = \min_{M,\, t \geq 1} \{|M| + \lceil \log t \rceil \mid M \text{ outputs } x \text{ in } t \text{ steps}\}\,. \qquad (2)$$

To provide intuition and give a concrete example of the usefulness of this time-bounded variant of Kolmogorov complexity to algorithms and complexity theory, we consider the following computational problem at the intersection of mathematics and computer science:

Explicit Construction of Primes: Given an integer $n \geq 2$, deterministically compute an n-bit prime number.[2]

[1]We formally define (time-bounded) Kolmogorov complexity in Section 2.

[2]For instance, the string x in Equation (1) is a 40-bit prime (733008047147 in decimal representation).

The fastest known algorithm that solves this problem runs in time $\widetilde{O}(2^{n/2})$ [LO87], and it is a longstanding open problem to improve this bound (see [TCH12]). Let $A(n)$ denote this procedure, and consider the sequence $\{p_n\}_{n \geq 2}$ of primes output by $A(n)$. Since we can encode the fixed algorithm A using $O(1)$ bits and any fixed number n using $O(\log n)$ bits, it follows that some program M of description length $O(\log n)$ runs in time $t = \widetilde{O}(2^{n/2})$ and prints p_n. Consequently, there is an n-bit prime p_n such that $\mathsf{Kt}(p_n) \leq O(1) + O(\log n) + \log t \leq n/2 + O(\log n)$. More generally, a faster algorithm yields improved bounds on the Kt complexity of some sequence of prime numbers. Conversely, it is possible to prove that if there is a sequence $\{q_n\}_{n \geq 2}$ of n-bit primes such that $\mathsf{Kt}(q_n) = \lambda_n$, then the problem of explicit constructing primes can be solved in time $\widetilde{O}(2^{\lambda_n})$.[3] This shows that one can completely capture the problem of explicitly constructing primes via time-bounded Kolmogorov complexity!

Note that in Kt complexity the time bound is not fixed and depends on the best possible description of x. In some contexts, it is desirable to restrict attention to programs M that run under a specified time bound $t(n)$, e.g., in time $\leq n^3$. This is captured by K^t complexity (see, e.g., [Sip83]), where $t \colon \mathbb{N} \to \mathbb{N}$ is a fixed function:

$$\mathsf{K}^t(x) = \min_M \{|M| \mid M \text{ outputs } x \text{ in } t(|x|) \text{ steps}\}. \tag{3}$$

As a recent application of time-bounded Kolmogorov complexity, Liu and Pass [LP20] connected one-way functions (OWF), a primitive that is essential to cryptography, to the computational difficulty of estimating the K^t complexity of an input string x, when t is a fixed polynomial. A bit more precisely, they showed that OWFs exist if and only if it is computationally hard on average to estimate $\mathsf{K}^t(x)$ for a random input string x (see their paper for the exact statement). This provides another striking example of the power and reach of time-bounded Kolmogorov complexity.

While connections between time-bounded Kolmogorov complexity and different areas of theoretical computer science have been known for a long time (see, e.g., [Sip83, Ko91, ABK+06, AF09]), recent applications of it to cryptography [LP20, RS21, LP21], learning [CIKK16, HN21], average-case complexity [Hir21], circuit complexity [OPS19], and proof search [Kra22] have led to much interest in this topic and to a number of related developments. We refer the reader to these papers and to [All92, All01, For04, Lee06, All17, LV19, All21] for more information on different time-bounded Kolmogorov complexity measures and their applications.

[3]As discovered by Levin, this is achieved by an algorithm that attempts to compute an n-bit prime by carefully simulating all programs of small description length for an appropriate number of steps until an n-bit prime is found.

Probabilistic (Time-Bounded) Kolmogorov Complexity. The need to use time-bounded Kolmogorov complexity in certain applications can create issues that are not present in the case of (time-unbounded) Kolmogorov complexity. More precisely, several central results from Kolmogorov complexity are not known to hold in a time-bounded setting. Some of them do survive under a plausible assumption (e.g., a *source coding theorem* holds for K^t under a strong derandomisation assumption [AF09]), but this leads to *conditional* results only. In other cases, the validity of a result in the setting of time-bounded Kolmogorov complexity is closely tied to a longstanding open problem in complexity theory (e.g., the computational difficulty of estimating $\mathsf{K}^t(x)$ and the aforementioned connection to OWFs [LP20]). We refer to [Lee06] for an extensive discussion on the similarities and differences between Kolmogorov complexity and its time-bounded counterparts.

Going beyond the technical difficulties of employing time-bounded Kolmogorov complexity, which some papers such as [Hir21] managed to overcome with the right assumptions in place, there is perhaps a more relevant issue in the application of notions such as Kt and K^t to algorithms and complexity: these classical measures refer to *deterministic* algorithms and programs. However, in many cases it is desirable or even necessary to consider *randomised* algorithms. Since the random strings that are part of the input of a randomised algorithm have high complexity, the classical theory of time-bounded Kolmogorov complexity might be inappropriate or simply cannot be applied in such contexts.

To mitigate these issues and develop a more robust theory of time-bounded Kolmogorov complexity that can be deployed in the important setting of randomised computations, some recent papers have explored *probabilistic* notions of time-bounded Kolmogorov complexity [Oli19, LO21, LOS21, GKLO22, LOZ22]. For this to make sense, we must conciliate the high complexity of a random string, which can be accessed by a randomised algorithm, with the goal of obtaining a succinct representation of $x \in \{0,1\}^*$. Note that simply storing a good choice of the random string r for a small program M that prints x when given r does not lead to a succinct representation of x.

The key concept employed in the aforementioned papers is that of a *probabilistic representation* of the string x. In other words, this is the code of a randomised program M such that, for most choices of its internal random string r, M prints x from r. Observe that the representation itself is a deterministic object: the code of M. However, to recover x from M, we must run the randomised algorithm M, meaning that we obtain x with high probability but there might be a small chance that M outputs a different string.[4] If $|M|$ is small, we obtain a succinct probabilistic representation of x. It is possible to introduce different variants of probabilistic time-bounded Kolmogorov complexity, and we properly define them in Section 3.

[4]This is similar to the notion of a pseudodeterministic algorithm from [GG11].

The investigation of probabilistic Kolmogorov complexity and of probabilistic representations is motivated from several angles:

(*i*) If we are running a randomised algorithm over an input string x, then storing a probabilistic representation of x instead of x can be done without loss of generality. There is already a small probability that the randomised algorithm outputs an incorrect answer, so it makes sense to tolerate a small probability of computing over a wrong input as well (i.e., when x is not correctly recovered from its probabilistic representation).

(*ii*) We will see later in the survey that probabilistic Kolmogorov complexity allows us in some cases to obtain *unconditional* versions of results that previously were only known to hold under strong complexity-theoretic assumptions.

(*iii*) As alluded to above, there are situations where the deterministic time-bounded measures simply cannot be applied due to the presence of randomised computations involving random strings of high complexity.

(*iv*) Finally, advances in probabilistic Kolmogorov complexity can be translated into results and insights for the classical notions of Kt complexity and K^t complexity, under certain derandomisation hypotheses.

Before describing our results and explaining the points mentioned above in more detail, we present a list of five fundamental questions to guide our investigation and exposition of probabilistic Kolmogorov complexity.

Q1. Usefulness: *Are there shorter probabilistic representations for natural objects, such as prime numbers? Can such representations detect structure in data that is inaccessible for* Kt *and* K^t*?*

Q2. Probabilistic Compression: *If succinct probabilistic representations exist, how can we efficiently compute one such representation? This is particularly relevant for data compression.*

Q3. Applications: *Are there interesting applications of probabilistic time-bounded Kolmogorov complexity to algorithms and complexity theory?*

Q4. Computational Hardness: *If provably secure cryptography exists, it must be impossible to efficiently detect certain patterns in data. Is it computationally hard to decide if a string admits a succinct probabilistic representation?*

Q5. Finding an Incompressible String: *Can we explicitly produce a string that does not admit a short probabilistic representation? What are such strings useful for?*

In the remaining parts of this article, we explain the recent progress on Questions Q1-Q5 achieved by references [Oli19, LO21, LOS21, GKLO22, LOZ22]. Along the way, we highlight some concrete open problems and present directions for further research. Due to space constraints, we often provide only a sketch of the underlying arguments, referring to the original references for more details.

Organisation and Overview. For convenience of the reader, we provide below a brief overview of each remaining section of this survey and how it relates to Questions Q1-Q5 described above.

– Section 2 fixes notation and formalises the deterministic time-bounded Kolmogorov complexity notions Kt and K^t.

– Section 3 formalises the intuitive concept of probabilistic representations discussed above. We introduce the probabilistic measures rKt, rK^t, and pK^t and describe some simple applications.

– Section 4 addresses Question Q1 (Usefulness) and explains a result from [LOS21] showing that infinitely many primes admit efficient probabilistic representations of sub-polynomial complexity. This is a significant improvement over the aforementioned $\approx n/2$ bound for Kt complexity.

– Section 5 covers the relation between sampling algorithms for a distribution over strings and the existence of probabilistic representations for individual strings [LO21, LOZ22]. Such results are called source coding theorems and have applications to Question Q2 (Probabilistic Compression).

– Section 6 approaches Question Q3 (Applications) and discusses applications of rKt, rK^t, and pK^t to average-case complexity and learning [GKLO22, LOZ22]. We employ these notions to simplify previous proofs, obtain new results that crucially rely on probabilistic Kolmogorov complexity, and establish unconditional analogues of theorems that were only known under derandomisation hypotheses.

– Section 7 is connected to Question Q1 (Usefulness) and focuses on the relation between time-bounded deterministic and probabilistic measures. We observe that these notions essentially coincide under strong enough derandomisation assumptions [Oli19, GKLO22]. Assuming them, insights from probabilistic Kolmogorov complexity readily translate into information about Kt and K^t.

– Section 8 sheds light on Question Q4 (Computational Hardness) by unconditionally establishing that certain computational problems about estimating the probabilistic time-bounded Kolmogorov complexity of an input string cannot be solved in probabilistic polynomial time [Oli19, LOS21].

– Section 9 shows that Question Q5 (Finding an Incompressible String) is closely related to the existence of hierarchy theorems for probabilistic time [LO21, LOS21], a fundamental question in computational complexity theory.

– Section 10 provides some concluding remarks and prospects for the potential impact of (probabilistic) time-bounded Kolmogorov complexity in algorithms and complexity.

Acknowledgements. We thank Michal Koucký for the invitation to write this survey. We are grateful to Eric Allender, Bruno P. Cavalar, Lijie Chen, Valentine Kabanets, Michal Koucký, Ninad Rajgopal, and Marius Zimand for sharing comments and suggestions on a preliminary version of the text. This work received support from the Royal Society University Research Fellowship URF\R1\191059 and from the EPSRC New Horizons Grant EP/V048201/1.

2 Preliminaries

For a positive integer m, we let $[m] \overset{\text{def}}{=} \{1, 2, \dots, m\}$. Given a non-negative real number α, we let $\lceil \alpha \rceil \in \mathbb{N}$ denote the smallest integer a such that $\alpha \leq a$. For a string $w \in \{0,1\}^*$, we use $|w| \in \mathbb{N}$ to denote its length. We let ϵ represent the empty string.

Let U be a Turing machine. For a function $t \colon \mathbb{N} \to \mathbb{N}$ and a string $x \in \{0,1\}^*$, we let

$$\mathsf{K}^t_U(x) \overset{\text{def}}{=} \min_{p \in \{0,1\}^*} \Big\{ |p| \;\mid\; U(p, \epsilon) \text{ outputs } x \text{ in at most } t(|x|) \text{ steps} \Big\}$$

be the *t-time-bounded Kolmogorov complexity* of x. The machine U is said to be *time-optimal* if for every machine M there exists a constant c_M such that for all $x \in \{0,1\}^n$ and $t \colon \mathbb{N} \to \mathbb{N}$ satisfying $t(n) \geq n$,

$$\mathsf{K}_U^{c_M \cdot t \log t}(x) \leq \mathsf{K}^t_M(x) + c_M,$$

where for simplicity we write $t = t(n)$. It is well known that there exist time-optimal machines (see, e.g., [LV19, Chapter 7]). We fix such a machine, and drop the index U when referring to time-bounded Kolmogorov complexity measures.

Given strings $x, y \in \{0,1\}^*$, we can also consider the *conditional t-time-bounded Kolmogorov complexity of x given y*, defined as

$$\mathsf{K}^t(x \mid y) \overset{\text{def}}{=} \min_{p \in \{0,1\}^*} \Big\{ |p| \;\mid\; U(p,y) \text{ outputs } x \text{ in at most } t(|x|) \text{ steps} \Big\}.$$

In the definitions above, the function $t \colon \mathbb{N} \to \mathbb{N}$ is fixed in advance. In many situations, it is also useful to consider a notion of time-bounded Kolmogorov complexity where the time bound of the machine is not fixed but instead affects the resulting complexity measure. One of the most prominent such measures is Levin's Kt complexity, defined as

$$\mathsf{Kt}(x) \overset{\text{def}}{=} \min_{p \in \{0,1\}^*,\, t \in \mathbb{N}} \Big\{ |p| + \lceil \log t \rceil \;\mid\; U(p,\epsilon) \text{ outputs } x \text{ in at most } t \text{ steps} \Big\}.$$

This definition can be extended to conditional Kt complexity $\mathsf{Kt}(x \mid y)$ in the natural way.

From now on, we will not distinguish between a Turing machine M and its encoding p_M according to U. While the running time t of M on an input y and the running time of the universal machine U on (p_M, y) might differ by a multiplicative factor of $O(\log t)$, this will be inessential in all results and applications discussed in this survey.[5]

We use $\mathsf{K}(x)$ to refer to the (time-unbounded) Kolmogorov complexity of the string x.

3 Probabilistic Notions of Kolmogorov Complexity: rKt, rK^t, and pK^t

In Section 2, we introduced two *deterministic* notions of time-bounded Kolmogorov complexity: K^t and Kt. In order to extend these definitions to the setting of *randomised* computations, we consider an algorithm with a short description that outputs a fixed string $x \in \{0,1\}^n$ with high probability. Intuitively, the code of this algorithm serves as a *probabilistic representation* of x.

A bit more formally, we consider a randomised Turing machine (RTM) M such that

$$\Pr_M[M(\epsilon) \text{ outputs } x] \geq 2/3.$$

Since we are interested in time-bounded representations, in our definitions we must decide if we require (1) $M(\epsilon)$ to run in time $\leq t$ over all computation paths;

[5]It is also possible to consider prefix-free notions of Kolmogorov complexity. Since our results hold up to additive $O(\log |x|)$ terms, we will not make an explicit distinction.

or (2) with probability $\geq 2/3$, $M(\epsilon)$ runs in time $\leq t$ and outputs x. It turns out that this distinction is not really crucial for the results discussed in this survey, since they are robust to additive overheads of order $\log n$. In more detail, by specifying and storing a positive integer $i \in [n]$, which can be represented using just $\log n$ bits, we can always enforce the machine M to stop in time 2^i.

Remark 1. In the definitions presented below, we abuse notation and refer to a machine M and its code. Formally, as in the definitions from the preceding section, M should be an arbitrary string (and not be restricted to a string that is a well-formed description of a machine) that is provided as input to the machine U.[6] This is important to guarantee that the Kolmogorov complexity of an arbitrary string of length n is at most $n + O(1)$. Defining Kolmogorov complexity and its time-bounded variants using the *code* of a machine might only allow us to prove an upper bound of $O(n)$, which can create issues in some applications where a tight worst-case bound is needed. To simplify the presentation, we blur this distinction in the remaining parts of this survey.

rK^t Complexity [BLvM05, LOS21].[7] This is the randomised analogue of K^t, where the time function $t\colon \mathbb{N} \to \mathbb{N}$ is fixed in advance. For a string $x \in \{0,1\}^*$, we let

$$\mathsf{rK}^t(x) \overset{\text{def}}{=} \min_{\text{RTM}\, M} \{|M| \mid M(\epsilon) \text{ outputs } x \text{ in } t(|x|) \text{ steps with probability} \geq 2/3\}$$

denote its *randomised t-time-bounded Kolmogorov complexity*. As an example of the use of rK^t, suppose a computationally unbounded party A holds a string x, and that A would like to communicate x to a t-time-bounded party B that has access to random bits. Then A can send $k = \mathsf{rK}^t(x)$ bits to B by communicating the description of a randomised Turing machine M as above. B is able to recover x from M with high probability simply by running $M(\epsilon)$.

pK^t Complexity [GKLO22]. Fix a function $t\colon \mathbb{N} \to \mathbb{N}$, as before. For a string $x \in \{0,1\}^*$, the *probabilistic t-time-bounded Kolmogorov complexity* of x is defined as

$$\mathsf{pK}^t(x) \overset{\text{def}}{=} \min \left\{ k \in \mathbb{N} \;\middle|\; \Pr_{w \sim \{0,1\}^{t(|x|)}} \Big[\exists\, \text{TM } M \in \{0,1\}^k,\ M(w) \text{ outputs } x \text{ within } t(|x|) \text{ steps} \Big] \geq \frac{2}{3} \right\}.$$

Note that M is a deterministic machine in the above definition. In other words, if $k = \mathsf{pK}^t(x)$, then with probability at least 2/3 over the choice of the random string w, given w the string x admits a t-time-bounded encoding of length k, i.e., $\mathsf{K}^t(x \mid w) \leq k$. In particular, if two parties share a typical *public* random string w,

[6]We assume that U has access to a tape with random bits.

[7][BLvM05] refers to this notion as CBP^t complexity.

then x can be transmitted with k bits and decompressed in time $t = t(|x|)$. For a reader familiar with standard complexity classes, the condition $\mathsf{K}^t(x) \leq s$ is reminiscent of NP, while $\mathsf{rK}^t(x) \leq s$ and $\mathsf{pK}^t(x) \leq s$ essentially correspond to MA and AM, respectively.

The definition of pK^t complexity is more subtle than the definitions of K^t and rK^t. In particular, small pK^t complexity provides a short efficient description only in the presence of a fixed, "good" random string. Interestingly, pK^t turns out to be surprisingly useful in applications of time-bounded Kolmogorov complexity, as discussed in Sections 5 and 6.

The following inequalities immediately follow from these definitions.

Fact 2. *For every string $x \in \{0,1\}^*$ and function $t \colon \mathbb{N} \to \mathbb{N}$, we have* $\mathsf{pK}^t(x) \leq \mathsf{rK}^t(x) \leq \mathsf{K}^t(x)$.

rKt Complexity [Oli19]. We can also consider the *randomised* Kt *complexity* of a string $x \in \{0,1\}^*$, defined as

$$\mathsf{rKt}(x) \overset{\text{def}}{=} \min_{\text{RTM } M,\, t \in \mathbb{N}} \left\{ |M| + \lceil \log t \rceil \mid M(\epsilon) \text{ outputs } x \text{ in } t \text{ steps with probability} \geq 2/3 \right\}.$$

All these probabilistic notions of time-bounded Kolmogorov complexity can be generalised to capture the conditional complexity of x given y in the natural way. As a concrete example, suppose a Boolean formula $F(x_1, \ldots, x_n)$ admits a satisfying assignment $\alpha \in \{0,1\}^n$ such that $\mathsf{rKt}(\alpha \mid F) \leq k$. Then we can find in time $\widetilde{O}(2^k \cdot |F|)$ and with probability $\geq 2/3$ a satisfying assignment of F by performing the following randomised computation: for each $i \in [k]$, enumerate all RTM M of description length i, run $M(F)$ for at most 2^{k-i} steps, and output the first string $\beta \in \{0,1\}^n$ generated in one of the simulations such that $F(\beta) = 1$.

An important property of Kolmogorov complexity is that, by a simple counting argument, most strings of length n are *incompressible*, i.e., they do not admit representations of length noticeably shorter than n. Similarly, most strings do not admit succinct probabilistic representations, even in the presence of a fixed advice string y.

Proposition 3 (Incompressibility)**.** *Let $n \geq 1$ and consider an arbitrary time bound $t(n)$. For each string $y \in \{0,1\}^*$, measure $C \in \{\mathsf{rK}^t, \mathsf{pK}^t, \mathsf{rKt}\}$, and integer $k \geq 1$, the following holds.*

$$\Pr_{x \sim \{0,1\}^n} [C(x \mid y) < n - k] = O(2^{-k}).$$

Proof Sketch. For $C \in \{\mathsf{rK}^t, \mathsf{rKt}\}$, the result follows from a simple counting argument, using that a valid probabilistic representation represents a single string (i.e., the success probability of printing the string is $\geq 2/3$, so it is uniquely specified given the machine).

On the other hand, when $C = \mathsf{pK}^t$, we argue as follows. If a large fraction of n-bit strings x have bounded pK^t complexity, by an averaging argument, there is a fixed choice of the random string $w \in \{0,1\}^{t(n)}$ such that, given w, a large fraction of the n-bit strings admit bounded descriptions for this choice of w as the random string. We can then use a similar counting argument to show that this is contradictory. See [GKLO22] for the details. $\square$

It is also possible to define pKt complexity, in analogy with the aforementioned definitions. However, since we are not aware of an interesting application of pKt, we will not discuss it here.

Other notions of time-bounded Kolmogorov complexity involving randomised computations have been considered in the literature. For instance, [BLvM05] considers CAM^t, a variant that combines randomness and nondeterminism. Due to space constraints, this survey will only cover rKt, rK^t, pK^t and their recent applications.

4 Prime Numbers with Short Descriptions and Pseudodeterministic PRGs

As briefly discussed in Section 1, an important question about prime numbers is whether they admit succinct representations, which is tightly connected to the fundamental problem of generating large primes deterministically. While this remains a notoriously difficult question to answer, we can still ask whether prime numbers admit succinct *probabilistic* representations. Results for this question were recently obtained in [OS17b, LOS21], by considering different notions of (time-bounded) randomised Kolmogorov complexity.

Before describing these results, we first note that it is impossible to compress *every* prime, given the Prime Number Theorem, which asserts that the number of primes whose values are less than or equal to N is roughly $N/\log N$. In particular, by a simple counting argument, this means that we cannot compress every n-bit prime to $o(n)$ bits. Therefore, here we ask whether there is an infinite sequence $\{p_m\}_{m\in\mathbb{N}}$ of increasing primes p_m that admit non-trivial probabilistic representations. The first non-trivial result of this form was established for rKt complexity.

Theorem 4 (rKt Upper Bounds for Primes [OS17b])**.** *For every $\varepsilon > 0$, there is an infinite sequence $\{p_m\}_{m\geq 1}$ of increasing primes p_m such that $\mathsf{rKt}(p_m) \leq |p_m|^{\varepsilon}$, where $|p_m|$ denotes the bit-length of p_m.*

Theorem 4 was proved via the construction of a *pseudodeterministic pseudorandom generator*. Informally, a pseudorandom generator (PRG) is an efficient procedure mapping a short string (called *seed*) to a long string, with the property that its output "looks random" to algorithms with bounded running time.[8] A PRG G is called *pseudodeterministic* if there is a probabilistic algorithm that, given a seed z, computes $G(z)$ with high probability. The following pseudodeterministic PRG was obtained in [OS17b].

Theorem 5 (A Pseudodeterministic Sub-Exponential Time PRG [OS17b]). *For every $\varepsilon > 0$ and $c, d \geq 1$, there exists a generator $G = \{G_n\}_{n\geq 1}$ with $G_n \colon \{0,1\}^{n^\varepsilon} \to \{0,1\}^n$ for which the following holds:*

Running Time: *There is a probabilistic algorithm that given n, $x \in \{0,1\}^{n^\varepsilon}$, runs in time $O\left(2^{n^\varepsilon}\right)$ and outputs $G_n(x)$ with probability $\geq 2/3$.*

Pseudorandomness: *For every algorithm A that runs in time at most n^c, there exist infinitely many input lengths n such that*

$$\left| \Pr_{x\sim\{0,1\}^n}[A(x) = 1] - \Pr_{z\sim\{0,1\}^{n^\varepsilon}}[A(G_n(z)) = 1] \right| \leq \frac{1}{n^d}.$$

Assuming Theorem 5, we show how to obtain Theorem 4.

Proof of Theorem 4. Let A be a deterministic polynomial-time algorithm for primality testing (e.g., [AKS02]), which takes as input an n-bit integer x and outputs 1 if and only if x is a prime. Suppose A runs in time n^c for some constant $c > 0$. Note that by the Prime Number Theorem, a uniformly random n-bit integer is a prime number with probability at least $1/O(n)$.

Let $\varepsilon > 0$ be any constant, and consider an infinitely often pseudodeterministic PRG $\{G_n\}_n$ from Theorem 5 with $G_n \colon \{0,1\}^{n^{\varepsilon/2}} \to \{0,1\}^n$ that is secure against (n^c)-time algorithms and has associated error parameter $\gamma = 1/n^2$. By the second item of Theorem 5, for infinitely many values of n, we have

$$\left| \mathbf{Pr}_{x\sim\{0,1\}^n}[A(x) = 1] - \mathbf{Pr}_{z\sim\{0,1\}^{n^{\varepsilon/2}}}[A(G_n(z)) = 1] \right| \leq \frac{1}{n^2},$$

which implies

$$\mathbf{Pr}_{z\sim\{0,1\}^{n^{\varepsilon/2}}}[A(G_n(z)) = 1] \geq \frac{1}{O(n)} - \frac{1}{n^2} \geq \frac{1}{O(n)}.$$

[8]Unconditionally constructing such PRGs is tightly connected to the derandomisation of probabilistic algorithms. While this remains a longstanding open problem, there has been progress in designing *pseudodeterministic* PRGs.

In particular, this means that there exists some $z \in \{0,1\}^{n^{\varepsilon/2}}$ such that $p := G(z)$ is an n-bit prime. By hardcoding n and this seed z, and using that $G(z)$ is a uniform procedure that can be computed probabilistically in time $t(n) = O\left(2^{n^{\varepsilon/2}}\right)$, we get that for infinitely many values of n, there is an n-bit prime p such that

$$\mathsf{rKt}(p) \le \left(n^{\varepsilon/2} + O(\log n) + O(1)\right) + \log\left(O\left(2^{n^{\varepsilon/2}}\right)\right) \le n^{\varepsilon},$$

as desired. □

For those primes shown to have small rKt complexity, given the corresponding encoding, one can probabilistically recover the prime in sub-exponential time. We can then further ask whether we can obtain succinct representations that can be decoded more efficiently, say, in polynomial time. Note that this is precisely to show that there are infinitely many primes whose $\mathsf{rK}^{\mathsf{poly}}$ complexity is small. This question was answered in the affirmative by a subsequent work of Lu, Oliveira and Santhanam.

Theorem 6 ($\mathsf{rK}^{\mathsf{poly}}$ Upper Bounds for Primes [LOS21])**.** *For every $\varepsilon > 0$, there is an infinite sequence $\{p_m\}_{m\ge1}$ of increasing primes p_m such that $\mathsf{rK}^t(p_m) \le |p_m|^{\varepsilon}$, where $t(n) = n^k$ for some constant $k = k(\varepsilon) \ge 1$, and $|p_m|$ denotes the bit-length of p_m.*

Similar to Theorem 4, Theorem 6 was proved via the construction of a certain pseudodeterministic PRG. Note that the reason why we got sub-exponential decoding time in Theorem 4 is due to the fact that the PRG from Theorem 5 requires sub-exponential time to compute. Then to obtain a polynomial decoding time as in Theorem 6, it suffices to construct a (pseudodeterministic) PRG that can be computed in polynomial time. Such a PRG was obtained in [LOS21] using a more sophisticated approach that builds on [OS17b].

Theorem 7 (A Pseudodeterministic Polynomial-Time PRG with 1 Bit of Advice [LOS21])**.**
For every $\varepsilon > 0$ and $c, d \ge 1$, there exists a generator $G = \{G_n\}_{n\ge1}$ with $G_n\colon \{0,1\}^{n^{\varepsilon}} \to \{0,1\}^n$ for which the following holds:

Running Time: *There is a probabilistic polynomial-time algorithm that given n, $x \in \{0,1\}^{n^{\varepsilon}}$, and an advice bit $\alpha(n) \in \{0,1\}$ that is independent of x, outputs $G_n(x)$ with probability $\ge 2/3$.*

Pseudorandomness: *For every algorithm A that runs in time at most n^c, there exist infinitely many input lengths n such that*

$$\left| \Pr_{x\sim\{0,1\}^n}[A(x) = 1] - \Pr_{z\sim\{0,1\}^{n^{\varepsilon}}}[A(G_n(z)) = 1] \right| \le \frac{1}{n^d}.$$

Using Theorem 7, it is easy to show Theorem 6 by mimicking the above proof of Theorem 4, with one caveat that computing the PRG in Theorem 7 requires one bit of advice. However, this extra bit can be hardcoded into the encoding without affecting its length by much.

We remark that the results presented above work in much more generality, and can be used to show that any dense language decidable in polynomial time admits infinitely many positive inputs of sub-polynomial $\mathsf{rKt}^{\mathsf{poly}}$ complexity. The set of primes is just one interesting example of such a language. We refer to [OS17b, LOS21] for additional applications of pseudodeterministic PRGs and for the proofs of Theorems 5 and 7.

We end this section with a couple of open problems. Note that both Theorem 4 and Theorem 6 show only that there are *infinitely many* values of n such that some n-bit prime has rKt or $\mathsf{rK}^{\mathsf{poly}}$ complexity at most n^{ε}.

Problem 8. *Show that for each $\varepsilon > 0$, there exists n_0 such that for every $n \geq n_0$, there is an n-bit prime p_n such that $\mathsf{rKt}(p_n) \leq n^{\varepsilon}$.*

Also, can we improve the sub-polynomial upper bounds to, say, poly-logarithmic?

Problem 9. *Prove that there is a constant $C \geq 1$ and an infinite sequence $\{p_m\}_{m\geq 1}$ of increasing primes p_m such that $\mathsf{rKt}(p_m) = (\log |p_m|)^C$.*

5 Sampling Algorithms, Coding Theorems, and Search-to-Decision Reductions

The coding theorem for Kolmogorov complexity roughly states that if a string x can be sampled with probability δ by some algorithm A, then its Kolmogorov complexity $\mathsf{K}(x)$ is at most $\log(1/\delta) + O_A(1)$. In particular, strings that can be generated with non-trivial probability by a program of small description length admit shorter representations. The coding theorem is a fundamental result in Kolmogorov complexity theory that has found many applications in theoretical computer science (see, e.g., [LV92, Lee06, Aar14, IRS21]). In fact, [Lee06] regards the coding theorem as one of the four pillars of Kolmogorov complexity.[9]

The proof of the coding theorem crucially explores the time-unbounded feature of the Kolmogorov complexity measure, and it is unclear how it can be extended to the time-bounded setting. Ideally, we would like to show that if a string x can be generated with probability δ by some *efficiently samplable* distribution, then its time-bounded Kolmogorov complexity $\mathsf{Kt}(x)$ is about $\log(1/\delta)$. One reason why such a time-bounded coding theorem is hopeful is that it can

[9]The other three are incompressibility, language compression, and symmetry of information.

be proven under certain strong derandomisation assumption [AF09].[10] In particular, under such an assumption, if a polynomial-time samplable distribution outputs a string x with probability at least δ, then $\mathsf{Kt}(x) \leq \log(1/\delta) + O(\log n)$. However, the latter result is only *conditional*, in the sense that it relies on an unproven assumption that seems far beyond the reach of currently known techniques. Moreover, strong assumptions of this form could even be false. While it remains unclear whether we can obtain a coding theorem for Kt, [LO21] considered the problem of establishing an *unconditional* coding theorem in the *randomised time-bounded setting*. Somewhat surprisingly, it can be shown *unconditionally* that if a string x can be sampled efficiently with probability δ, then $\mathsf{rKt}(x) \leq O(\log 1/\delta) + O(\log n)$. In a subsequent work [LOZ22], this result is further improved to $\mathsf{rKt}(x) \leq (2 + o(1)) \cdot \log 1/\delta + O(\log n)$.

Theorem 10 (Coding Theorem for rKt [LOZ22])**.** *Suppose there is an efficient algorithm A for sampling strings such that $A(1^n)$ outputs a string $x \in \{0,1\}^n$ with probability at least δ. Then*

$$\mathsf{rKt}(x) \leq 2\log(1/\delta) + O\Big(\log n + \log^2 \log(1/\delta)\Big),$$

where the constant behind the $O(\cdot)$ depends on A and is independent of the remaining parameters. Moreover, given x, the code of A, and δ, it is possible to compute in time $\mathsf{poly}(n, |A|)$, with probability ≥ 0.99, a probabilistic representation of x that satisfies this rKt-complexity bound. (*The running time of this algorithm does not depend on the time complexity of A.*)

Similar to the results in the previous section that are concerned with the compressibility of prime numbers, the results of [LO21, LOZ22] again show the power of utilizing randomness in Kolmogorov complexity, which enables us to establish results for time-bounded Kolmogorov complexity that seem very difficult to show in the deterministic setting. We refer to these papers for a discussion of the techniques employed to show an unconditional coding theorem for rKt.

We note that (as in previous work of [LO21]) the coding theorem in Theorem 10 has an unexpected *constructive* feature: it gives a polynomial-time probabilistic algorithm that, when given x, the code of the sampler, and δ, outputs a probabilistic representation of x that certifies the claimed rKt complexity bound. (Additionally, the running time of this algorithm does not depend on the running time of the sampler.) Such an *efficient* coding theorem has interesting implications for search-to-decision reductions for rKt. Recall that a search-to-decision reduction is an efficient procedure that allows one to find solutions to a problem from the

[10]The assumption in [AF09] states that there is a language $L \in \mathsf{TIME}\big[2^{O(n)}\big]$ that requires Boolean circuits of size $2^{\Omega(n)}$ for all but finitely many n, even in the presence of oracle gates to a Σ_2^p-complete problem in the circuit.

mere ability to decide when a solution exists. Using results from [LO21, LOZ22], one can show the following search-to-decision reduction for rKt.

Theorem 11 (Instance-Wise Search-to-Decision Reduction for rKt [LO21])**.** *Let $\mathcal{O}$ be a function that linearly approximates rKt complexity. That is, for every $x \in \{0,1\}^*$,*

$$\Omega(\mathsf{rKt}(x)) \;\leq\; \mathcal{O}(x) \;\leq\; O(\mathsf{rKt}(x)).$$

Then there is a randomised polynomial-time algorithm with access to $\mathcal{O}$ that, when given an input string x, outputs with probability ≥ 0.99 a valid rKt representation of x of complexity $O(\mathsf{rKt}(x))$. Furthermore, this algorithm makes a single query q to $\mathcal{O}$, where $q = x$.

Proof Sketch. We would like to invoke Theorem 10 to efficiently compute an rKt representation of x, but the "moreover" part of this result requires the explicit code of a sampler. The idea is to construct a "universal" sampler that outputs x with the desired probability, then to hit this sampler with an appropriate coding theorem for rKt. For simplicity, suppose we knew the exact value $k = \mathsf{rKt}(x) \in \mathbb{N}$. Consider the following sampler A:

$A(1^n)$: Randomly selects a randomised program M of length k among all strings in $\{0,1\}^k$. Run M for at most 2^k steps, then output the n-bit string that M outputs during this simulation (or the string 0^n if M does not stop or its output is not an n-bit string).

Note that A runs in time $t = \mathsf{poly}(n, 2^k) = \mathsf{poly}(2^k)$ (since $k \geq \log n$ for any n-bit string), and that it outputs x with probability at least $\delta = 2^{-k} \cdot 2/3$, since by the definition of k at least one such program prints x with probability at least $2/3$. By the coding theorem for rKt from [LO21] (which is stated in a slightly more general form than Theorem 10), one obtains that $\mathsf{rKt}(x) = O(\log(1/\delta)) + O(\log t) = O(k)$. Since A is an explicit algorithm, crucially, its "moreover" part implies that we can efficiently output an rKt-representation of x of complexity $O(k)$. This completes the sketch of the proof.

We refer to [LO21, Section 4] for the formal proof of Theorem 11, which is a simple adaptation of the idea described here. □

An interesting feature of the above search-to-decision reduction is that it is *instance-wise* in the sense that to produce a near-optimal rKt representation of x, we only need to make a single query to a decision oracle for rKt on the same x.[11] Note that there are known search-to-decision reductions in the context of time-bounded Kolmogorov complexity with respect to various notions of complexity

[11] This is also called a *search-to-profile* reduction in some references in Kolmogorov complexity [RSZ21].

(e.g., [CIKK16, Hir18, Ila20, ILO20, LP20, Ila21]), but they require an oracle to the decision problem that is correct on all or at least on a large fraction of inputs. As a consequence of this feature, we can easily derive the following result.[12]

Corollary 12 ("Short Lists with Short Programs" [LO21])**.** *Given a string x of length n, it is possible to compute with probability ≥ 0.99 and in polynomial time a collection of at most $\ell = \log(n)$ strings $M_1, \ldots, M_\ell$ such that at least one of these strings is a valid rKt representation of x of complexity $O(\mathsf{rKt}(x))$.*

Proof. We run the instance-wise search-to-decision reduction on the input x. While it is not clear how to efficiently estimate $\mathsf{rKt}(x)$, we can still "guess" the rKt complexity of x to be of order 2^i, for each $i \in \{1, 2, \ldots, \log n\}$. We run the procedure on each possible guess, obtaining a list of strings $M_1, \ldots, M_\ell$, where $\ell = \log n$. Since there is at least one value i such that $2^i = \Theta(\mathsf{rKt}(x))$, we have the guarantee that in this case the reduction outputs with probability at least 0.99 a valid rKt representation of x of similar complexity. Therefore, the list contains with probability at least 0.99 a representation of the desired form. $\square$

While the above coding theorem for rKt is a novel development after a long gap in an area with only conditional results, it has an important drawback: the rKt upper bound is at least $2\log(1/\delta))$ and hence is *sub-optimal*. In contrast, the bounds in the time-unbounded setting and in the conditional result of [AF09] mentioned above have the form $\log(1/\delta)$. A natural question then is whether we can show a coding theorem for rKt with an optimal dependence on the probability parameter δ, which is crucial in many applications of the result. It turns out that under a certain hypothesis about the security of cryptographic pseudorandom generators[13], the rKt bound in Theorem 10 is essentially optimal if we consider only coding theorems that are *efficient*, i.e., where an rKt representation can be constructed in polynomial time regardless of the running time of the sampler. In particular, [LOZ22] showed that in this case, there is no efficient coding theorem that can achieve a bound of the form $\mathsf{rKt}(x) \leq (2 - o(1)) \cdot \log(1/\delta) + \mathrm{poly}(\log n)$. On the other hand, the conditional coding theorem for Kt in [AF09] is not efficient. This leads to the following open problem on (unconditionally) showing an *existential* coding theorem for rKt with optimal parameters.

[12]Results of this form were previously known in time-unbounded Kolmogorov complexity (see [BMVZ18]).

[13]The hypothesis states that there is a pseudorandom generator $G\colon \{0,1\}^{\ell(n)} \to \{0,1\}^n$, where $(\log n)^{\omega(1)} \leq \ell(n) \leq n/2$, computable in time $\mathrm{poly}(n)$ that is secure against *uniform algorithms* running in time $2^{(1-\Omega(1))\cdot \ell(n)}$. Note that every candidate PRG of seed length $\ell(n)$ can be broken in time $2^{\ell(n)} \cdot \mathsf{poly}(n)$ by trying all possible seeds. This hypothesis can be viewed as a cryptographic analogue of the well-known strong exponential time hypothesis (SETH) about the complexity of k-CNF SAT [IP01].

Problem 13. *Show that if there is an efficient algorithm A for sampling strings such that $A(1^n)$ outputs a string $x \in \{0,1\}^n$ with probability at least δ, then* $\mathsf{rKt}(x) \leq \log(1/\delta) + \mathrm{poly}(\log n)$.

To this point, we have mentioned the existence of an optimal coding theorem for time-unbounded Kolmogorov complexity and an optimal conditional coding theorem for Kt (in fact, the conditional result holds even for K^t for $t = \mathsf{poly}(n)$). Also, an unconditional coding theorem can be obtained for rKt but its dependency on the probability parameter δ is not $\log(1/\delta)$ (Theorem 10). Note that rKt can be viewed as a "relaxed" notion of Kt and is intermediate between K and Kt. If we consider some further relaxed notion of time-bounded Kolmogorov complexity, can we show a coding theorem that is both unconditional and optimal?

Note that the time-bounded measure pK can be viewed as an intermediate notion between time-unbounded Kolmogorov complexity and time-bounded rK. It turns out that pK^t admits an optimal coding theorem.

Theorem 14 (Coding Theorem for pK^t [LOZ22])**.** *Suppose there is a randomised algorithm A for sampling strings such that $A(1^n)$ runs in time $T(n) \geq n$ and outputs a string $x \in \{0,1\}^n$ with probability at least $\delta > 0$. Then*

$$\mathsf{pK}^t(x) = \log(1/\delta) + O(\log T(n)),$$

where $t(n) = \mathrm{poly}(T(n))$ and the constant behind the $O(\cdot)$ depends on $|A|$ and is independent of the remaining parameters.

The proof of Theorem 14 is similar in spirit to that of the conditional coding theorem for $\mathsf{K}^{\mathrm{poly}}$ in [AF09]. As an application of the latter, [AF09] showed a *conditional* characterisation of the worst-case running times of languages that are in average polynomial time over all samplable distributions. Using Theorem 14, [LOZ22] provided an *unconditional* characterisation, and this will be discussed in Section 6.

Finally, we can connect the time-bounded coding theorems discussed in this section to the compressibility of prime numbers discussed in the previous section, via the following equivalence.

Theorem 15 (Equivalence Between Samplability and Compressibility [LO21])**.** *Let $\delta \colon \mathbb{N} \to [0,1]$ be a time-constructible function. The following statements are equivalent.*

(i) **Samplability.** *There is a randomised algorithm A for sampling strings such that, for infinitely many (resp. all but finitely many) n, $A(1^n)$ runs in time $(1/\delta(n))^{O(1)}$ and outputs an n-bit prime q_n with probability at least $\delta(n)^{O(1)}$.*

(*ii*) **Compressibility.** *For infinitely many (resp. all but finitely many) n, there is an n-bit prime p_n with* $\mathsf{rKt}(p_n) = O(\log(1/\delta(n)))$.

Proof Sketch. The implication from (*i*) to (*ii*) relies on the existing coding theorem for rKt. The other direction employs a universal sampler in the spirit of the proof of Theorem 11 sketched above. See [LO21] for the details. $\square$

Theorem 15 can be seen as an analogue of the relation between deterministically constructing large primes and obtaining Kt upper bounds for primes, which was explained in Section 1. Using this result, the problem of showing that prime numbers have smaller rKt complexity (Problem 9) can be reduced to showing the existence of a faster sampling algorithm for primes. In particular, if we can sample an n-bit prime p_n in time $2^{\mathrm{poly}(\log n)}$ with probability at least $2^{-\mathrm{poly}(\log n)}$, then $\mathsf{rKt}(p_n) \leq \mathrm{poly}(\log n)$.

We remark that an even tighter equivalence between samplability and compressibility can be established using pK^t complexity, thanks to the optimality of Theorem 14.

6 Applications to Average-Case Complexity and Learning Theory

Understanding the relation between the average-case complexity of NP and its worst-case complexity is a central problem in complexity theory. More concretely, if every problem in NP is easy to solve on average, can we solve NP problems in polynomial time in the worst case? While addressing this question remains a longstanding open problem, significant results have been achieved in recent years using techniques from time-bounded Kolmogorov complexity [Hir20a, Hir21, CHV22] (see [Hir22a] for an overview). Related techniques have also led to the design of faster learning algorithms under the assumption that NP is easy on average [HN21]. Interestingly, the problems investigated in these references make no reference to Kolmogorov complexity. Still, the corresponding proofs rely on K^t complexity and its properties in important ways.

In this section, we describe recent applications of pK^t complexity to average-case complexity and learning theory [GKLO22, LOZ22]. While the definition of pK^t is more subtle compared with K^t and rK^t, its use comes with important benefits. As we explain later in this section, depending on the context, pK^t complexity allows us to extend previous results to the important setting of randomised computations, significantly simplify an existing proof, or obtain an unconditional result.

Average-Case Complexity. We first review some standard definitions from average-case complexity theory (see [BT06] for a survey of this area). Recall that $D =$

$\{D_n\}_{n \geq 1}$, where each D_n is a distribution supported over $\{0, 1\}^*$, is called an ensemble of distributions. We say that $D \in \mathsf{PSamp}$ (or D is P-*samplable*) if there is a randomised polynomial-time algorithm A such that, for every $n \geq 1$, $A(1^n)$ is distributed according to D_n.

Let D be an ensemble of distributions. We say that a language L is solvable in *polynomial time on average* with respect to D if there is a deterministic algorithm A such that, for every n and for every x in the support of D_n, $A(x; n) = L(x)$, and there is a constant $\varepsilon > 0$ such that $\mathbf{E}_{x \sim D_n}[t_{A,n}(x)^{\varepsilon}/n] = O(1)$, where $t_{A,n}(x)$ denotes the running time of A on input $(x; n)$. We remark that this is equivalent to the existence of a deterministic algorithm B and of a polynomial p such that the following conditions hold:

- For every n, $\delta > 0$, and string x in the support of D_n, $B(x; n, \delta)$ outputs either $L(x)$ or the failure symbol $\perp$;
- For every n, $\delta > 0$, and every string x in the support of D_n, $B(x; n, \delta)$ runs in time at most $p(n, 1/\delta)$;
- For every n and every $\delta > 0$,

$$\Pr_{x \sim D_n}[B(x; n, \delta) = \perp] \leq \delta.$$

We refer to [BT06] for more information about this definition and its motivation.

A pair (L, D) is a *distributional problem* if $L \subseteq \{0, 1\}^*$ and D is an ensemble of distributions. For a complexity class $\mathfrak{C}$ (e.g., $\mathfrak{C} = \mathsf{NP}$), we let $\mathsf{Dist}\mathfrak{C}$ denote the set of distributional problems (L, D) with $L \in \mathfrak{C}$ and $D \in \mathsf{PSamp}$. We say that $(L, D) \in \mathsf{AvgP}$ if L is solvable in polynomial time on average with respect to D.

Note that in the equivalent definition of AvgP the deterministic algorithm is never incorrect on an input x in the support of the distribution. Similarly, it is possible to consider average-case complexity with respect to *randomised errorless heuristic schemes*. Roughly speaking, such randomised algorithms are allowed to sometimes output the wrong answer, provided that on every input x in the support of the distribution, the fraction of random strings for which the algorithm outputs the wrong answer is small compared to the fraction of random strings for which it outputs either the right answer or the fail symbol $\perp$. Analogously to the definition of AvgP, if a distributional problem (L, D) admits a randomised errorless heuristic scheme, we say that $(L, D) \in \mathsf{AvgBPP}$. We refer again to [BT06] for the precise definition of this class and for an extensive discussion of this notion and its extensions.[14]

[14] It is also possible to consider randomised algorithms that can sometimes be incorrect on an input x with high probability over their internal randomness. This leads to the class $\mathsf{HeurBPP}$ of distributional problems. Relaxing some assumptions in this section to the setting of $\mathsf{HeurBPP}$ is an interesting research direction (see, e.g., [HS22]).

6.1 Worst-Case Time Bounds for Average-Case Easy Problems

Suppose that a language L is average-case easy. That is, L is solvable in deterministic polynomial time on average with respect to *all* P-samplable distributions. What can we say about the time needed to solve L in the *worst case*? In a beautiful work, Antunes and Fortnow [AF09] *characterised* the worst-case running time of such a language using the notion of *computational depth* [AFvM01]. Here the computational depth of a string x for a time bound t is defined as the difference $\mathsf{K}^t(x) - \mathsf{K}(x)$. It was shown in [AF09], under a strong derandomisation assumption, that a language L is average-case easy *if and only if* it can be solved in time $2^{O(\mathsf{K}^{\mathrm{poly}}(x)-\mathsf{K}(x)+\log(|x|))}$ for every input $x \in \{0,1\}^*$. The proof of this result crucially relied on the use of an *optimal* coding theorem for K^t. Since such a coding theorem is only known under a strong derandomisation assumption (see Section 5), the aforementioned characterisation is subject to the same unproven assumption.

As also mentioned in Section 5, it was observed in [LOZ22] that an optimal coding theorem can be unconditionally proved for pK^t (Theorem 14). It turns out that such a coding theorem enables us to show an *unconditional* version of Antunes and Fortnow's characterisation, where the worst-case running times for languages that are average-case easy can be characterised using a notion of *probabilistic computational depth*.

A key idea in the proof of this result is a notion of *universal* distribution via pK^t. More specifically, for a computable time bound function t, we define m^t to be the (semi-)distribution whose probability density function is $\mathsf{m}^t(x) \overset{\mathrm{def}}{=} 2^{-\mathsf{pK}^t(x)-b\log|x|}$, where $b > 0$ is a large enough constant (that depends only on t).[15]

Theorem 16 (Unconditional "Worst-Case Time Bounds for Average-Case Easy Problems" [LOZ22]). *The following conditions are equivalent for any language $L \subseteq \{0,1\}^*$.*[16]

1. *For every P-samplable distribution D, L can be solved in polynomial time on average with respect to D.*

2. *For every polynomial p, L can be solved in polynomial time on average with respect to m^p.*

[15]The reason why we define $\mathsf{m}^t(x)$ this way instead of using just $2^{-\mathsf{pK}^t(x)}$ is to make sure that it forms a (semi-)distribution, i.e., that the sum of the probabilities is at most 1. More specifically, for every t, there is some constant $b > 0$ such that $\mathsf{K}(x) \leq \mathsf{pK}^t(x) + b\log|x|$ for every x (see [LOZ22, Lemma 32]), so $\sum_{x\in\{0,1\}^*} 2^{-\mathsf{pK}^t(x)-b\log|x|} \leq \sum_{x\in\{0,1\}^*} 2^{-\mathsf{K}(x)} \leq 1$, where the second inequality follows from Kraft's inequality. (Formally, to apply Kraft's inequality we need to consider prefix-free encodings. This is not an issue here, as a large enough constant b makes this possible.)

[16]In this statement and in its proof, we do not make a distinction between distributions and semi-distributions. (In a semi-distribution, the sum of the probabilities might add up to less than 1.)

3. *For every polynomial p, there exists a constant $c > 0$ such that the running time of some algorithm that computes L is bounded by $2^{O\left(\mathsf{pK}^p(x) - \mathsf{K}(x) + c\log(|x|)\right)}$ for every input $x \in \{0,1\}^*$.*

Proof Sketch. For simplicity, to sketch the proof of this theorem we will also consider the notion of average-case easiness with respect to *single distributions* instead of ensembles of distributions,[17] which does not incur a loss of generality (see [BT06, Section 6]).

We first sketch the equivalence between Item 1 and Item 2. We need to show that the class of distributions m^{poly} is "universal" for the class of P-samplable distributions, in the sense that a language L is polynomial-time on average with respect to m^{poly} if and only if the same holds with respect to all P-samplable distributions. Recall that if a distribution D *dominates* another distribution D' (i.e., $D(x) \gtrsim D'(x)$ for all x) and L is polynomial-time on average with respect to D, then the same holds with respect to D'. Therefore, to show the "universality" of m^{poly}, it suffices to establish the following claims.

1. Every P-samplable distribution is dominated by m^p, for some polynomial p.

2. For every polynomial p, m^p is dominated by some P-samplable distribution.

The first item above says that for every P-samplable D, $\mathsf{m}^p(x) \gtrsim D(x)$ for some polynomial p, which, by the definition of m^p, means $\mathsf{pK}^p(x) \lesssim \log(1/D(x))$. Note that this is essentially an optimal coding theorem for $\mathsf{pK}^{\text{poly}}$ and hence follows from Theorem 14. To see the second item, consider any polynomial p. We define a P-samplable distribution roughly as follows. We first pick n with probability $\frac{1}{n\cdot(n+1)}$, and then randomly pick $k \in [2n]$, $w \in \{0,1\}^{p(n)}$, and a program $M \in \{0,1\}^k$. We then run $M(w)$ for at most $p(n)$ steps and output the string that M outputs. It is easy to see that for every $x \in \{0,1\}^n$, the above sampling process outputs x with probability at least $2^{-\mathsf{pK}^p(x)}/n^{O(1)}$ and hence dominates m^p.

It remains to show the equivalence between Item 2 and Item 3. Here we describe the implication from Item 2 to Item 3, which highlights the use of a fundamental result in Kolmogorov complexity called *Language Compression.* The other direction follows from a simple calculation (see [LOZ22]).

Consider the time bound t described by an arbitrary polynomial p. Let A be an algorithm that solves L in polynomial time on average with respect to m^t, and let $t_A(x)$ denote the running time of A on input x. For $n, i, j \in \mathbb{N}$ with $i, j \le n^2$, define

$$S_{i,j,n} \overset{\text{def}}{=} \left\{x \in \{0,1\}^n \mid 2^i \le t_A(x) \le 2^{i+1} \text{ and } \mathsf{pK}^t(x) + b\log|x| = j\right\}.$$

[17]An algorithm A runs in polynomial time on average with respect to a (semi-)distribution D if there exists a constant ε such that, $\sum_{x\in\{0,1\}^*} \frac{t_A(x)^\varepsilon}{|x|} D(x) \le O(1)$, where $t_A(x)$ denotes the running time of A on input x.

Consider a nonempty set $S_{i,j,n}$, and let $r \in \mathbb{N}$ be such that $2^r \le \left|S_{i,j,n}\right| < 2^{r+1}$. We claim that for every $x \in S_{i,j,n}$, its (time-unbounded) Kolmogorov complexity

$$\mathsf{K}(x) \le r + O(\log n). \tag{4}$$

To see this, note that given i, j, n, we can first enumerate all the elements in $S_{i,j,n}$, which can be done since t is computable, and then using additional $r + 1$ bits, we can specify x in $S_{i,j,n}$. We remark that the core idea behind the above argument is the language compression theorem for (time-unbounded) Kolmogorov complexity, which states that for every (computable) language L, $\mathsf{K}(x) \le \log |L \cap \{0,1\}^n| + O(\log n)$ for all $x \in L \cap \{0,1\}^n$.[18]

Now fix any n and $i, j \le n^2$. Let r be such that $2^r \le \left|S_{i,j,n}\right| < 2^{r+1}$. Then by assumption and by the definition of $S_{i,j,n}$, we have for some constants $\varepsilon, d > 0$,

$$d \ge \sum_{x \in S_{i,j,n}} \frac{t_A(x)^\varepsilon}{|x|} \cdot \mathsf{m}^t(x) \ge 2^r \cdot \frac{2^{\varepsilon \cdot i}}{n} \cdot 2^{-j} = 2^{\varepsilon \cdot i + r - j - \log n},$$

which yields $\varepsilon \cdot i + r - j - \log n \le \log d$. By Equation (4) and using $j = \mathsf{pK}^t(x) + b \log n$, this implies that for every $x \in S_{i,j,n}$,

$$\varepsilon \cdot i \le \mathsf{pK}^t(x) - \mathsf{K}(x) + O(\log n).$$

Therefore, we have that for every $x \in S_{i,j,n}$,

$$t_A(x) \le 2^{i+1} \le 2^{\varepsilon^{-1} \cdot \left(\mathsf{pK}^t(x) - \mathsf{K}(x) + O(\log n)\right)} = 2^{O\left(\mathsf{pK}^t(x) - \mathsf{K}(x) + c \log(|x|)\right)},$$

where $c > 0$ is a large enough constant independent of $n = |x|$. Since it is not hard to see that every $x \in \{0,1\}^n$ is in some set $S_{i,j,n}$, the result follows. □

6.2 Probabilistic Average-Case Easiness Implies Worst-Case Upper Bounds

The section covers recent developments from [GKLO22], which build on the breakthrough results of [Hir21] and on the subsequent papers [CHV22, GK22, Hir22b]. In short, the results from [Hir21] hold in the setting of *deterministic* computations, while [GKLO22] provides a framework that allows new relations between average-case complexity and worst-case complexity to be established in the more robust setting of *randomised* computations.

[18] In fact, it is possible to slightly modify the above argument, by appropriately defining a language with slices in correspondence to the sets $S_{i,j,n}$, so that language compression can be applied directly.

Next, we provide a high-level exposition of some results from [GKLO22] and their proofs. In particular, we explain the role of (conditional) versions of "language compression" and "symmetry of information" for pK^t, and how pK^t turns out to be a complexity measure that is particularly well-suited for these applications (see Remark 21 on "Why pK^t complexity?").

Our goal is to show a *worst-case* complexity upper bound for an arbitrary language $L \in \mathsf{NP}$ under an *average-case* easiness assumption, such as $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ or the weaker $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$. Note that Theorem 16 naturally suggests an approach: if L is easy on average (Item 1), then we can compute L on every input $x \in \{0,1\}^*$ (Item 3) in time

$$2^{O(\mathsf{pK}^p(x) - \mathsf{K}(x))} \cdot \mathsf{poly}(|x|),$$

where $p(\cdot)$ is a fixed but arbitrary polynomial. Therefore, if we could show that the quantity $\mathsf{pK}^p(x) - \mathsf{K}(x)$ is bounded for *every* x, we would be done. (Note that this is indeed the case for a uniformly *random* x, since $\mathsf{pK}^t(x)$ and $\mathsf{K}(x)$ are close to $n = |x|$ with high probability.)

This is not possible, but we can still hope to adapt the proof of Theorem 16 to obtain a more useful bound, under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$. A closer inspection of the argument reveals that the value $\mathsf{K}(x)$ in the bound $\mathsf{pK}^p(x) - \mathsf{K}(x)$ comes from the use of *language compression* for (time-unbounded) Kolmogorov complexity, which is applied to the sets $S_{i,j,n}$. If we had a language compression theorem for a time-bounded measure γ (e.g., $\gamma = \mathsf{K}^t$), we would be able to derive a worst-case running time exponent of the form $\mathsf{pK}^p(x) - \gamma(x)$. This makes progress towards our goal, since $\gamma(x) \geq \mathsf{K}(x)$. This initial idea turns out to be feasible, for $\gamma = \mathsf{pK}^q$ (think of $q(\cdot)$ as a polynomial larger than $p(\cdot)$).

Theorem 17 (Language Compression for pK^t under $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$; Informal[19])**.** *If* $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$*, then for every language* $S \in \mathsf{AM}$*, there is a polynomial* q *such that for every* $x \in S \cap \{0,1\}^n$*,*

$$\mathsf{pK}^q(x) \leq \log |S \cap \{0,1\}^n| + \log q(n).$$

In order to implement the aforementioned plan, we need to make sure that the sets $S_{i,j,n}$ provide a language S that is *easy to compute*, since this is an assumption in Theorem 17. One can sidestep this issue by settling for a weaker result which assumes that the running time t_A of the average-case algorithm on a given input can be efficiently estimated without running the algorithm. This notion leads to a class of distributional problems called $\mathsf{Avg}_{\mathsf{BPP}}\mathsf{BPP}$ in [GKLO22], and to the stronger

[19]For technical reasons, the actual formulation of this result considers an ensemble of promise problems with padded inputs of the form $(x, 1^m)$, where $|x| = \ell(m)$. For simplicity, we omit this here. See [GKLO22] for the precise statement.

initial assumption that $\mathsf{DistNP} \subseteq \mathsf{Avg}_{\mathsf{BPP}}\mathsf{BPP}$. Another crucial idea, which we will not cover in more detail here, is to prove that $\mathsf{pK}^t(y)$ can be efficiently estimated for every string y under the assumption that NP is easy on average. We can then apply (an extension of) Theorem 17 to appropriately modified sets $S'_{i,j,n}$, which yields a worst-case running time of the form

$$2^{O(\mathsf{pK}^p(x)-\mathsf{pK}^q(x))} \cdot \mathsf{poly}(|x|).$$

One could hope for the quantity $\mathsf{pK}^p(x)-\mathsf{pK}^q(x)$, called the (p,q)-*probabilistic computational depth* of x, to be bounded for every string x. While this is not clear for the polynomials $p(n)$ and $q(n)$, a simple but neat argument involving a telescoping sum [Hir21, GKLO22] shows that, for any string x of length n, for some time bound $t(n) \leq 2^{O(n/\log n)}$ we have $\mathsf{pK}^t(x)-\mathsf{pK}^{\mathrm{poly}(t)}(x) = O(n/\log n)$. Intuitively, if we could adapt the previous strategy so that it yields more general worst-case upper bounds involving $(t, \mathrm{poly}(t))$-*probabilistic computational depth*, then a non-trivial exponent of $O(n/\log n)$ would be achieved by applying the argument to each choice of $t \leq 2^{O(n/\log n)}$.

A careful implementation of this plan leads to the following stronger consequence, where the worst-case upper bound holds for any language $L \in \mathsf{AM}$.

Theorem 18 ([GKLO22]). *If* $\mathsf{DistNP} \subseteq \mathsf{Avg}_{\mathsf{BPP}}\mathsf{BPP}$*, then* $\mathsf{AM} \subseteq \mathsf{BPTIME}[2^{O(n/\log n)}]$.

Can we obtain a similar worst-case upper bound under the weaker and more natural assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$? (In other words, without assuming that the running time of the average-case algorithm can be efficiently estimated?) This is currently open. However, it is possible to prove the following implications, which can be seen as a strengthening of some results from [Hir21] to the randomised setting. Recall that UP denotes the set of languages in NP whose positive instances admit unique witnesses.

Theorem 19 (Probabilistic Worst-Case to Average-Case Reductions [GKLO22]). *The following results hold.*

1. *If* $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$*, then* $\mathsf{UP} \subseteq \mathsf{RTIME}\left[2^{O(n/\log n)}\right]$.
2. *If* $\mathsf{Dist}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{AvgBPP}$*, then* $\mathsf{AM} \subseteq \mathsf{BPTIME}\left[2^{O(n/\log n)}\right]$.
3. *If* $\mathsf{DistPH} \subseteq \mathsf{AvgBPP}$*, then* $\mathsf{PH} \subseteq \mathsf{BPTIME}\left[2^{O(n/\log n)}\right]$.

The proof of Theorem 19 relies on *Symmetry of Information*, another pillar of Kolmogorov complexity (see [Lee06]). To describe a pair (x, y) of strings, one can combine the most succinct representation of x with the most succinct representation of y when x is given as advice. In Kolmogorov complexity, this is captured by the inequality $\mathsf{K}(x, y) \leq \mathsf{K}(x) + \mathsf{K}(y \mid x) + O(\log(|x| + |y|))$. The symmetry of information principle is a theorem in Kolmogorov complexity stating that this is essentially the most economical way of describing the pair (x, y). In other words: $\mathsf{K}(x, y) \geq \mathsf{K}(x) + \mathsf{K}(y \mid x) - O(\log(|x| + |y|))$. One can then easily derive that $\mathsf{K}(x) - \mathsf{K}(x|y) = \mathsf{K}(y) - \mathsf{K}(y \mid x)$, up to a term of order $O(\log(|x| + |y|))$. Roughly speaking, the information that x contains about y is about the same the information that y contains about x.

The proof of symmetry of information for K requires an exhaustive search, which is not available in the time-bounded setting. Nevertheless, different forms of the principle can still be established in this more delicate setting under average-case easiness assumptions [GK22, Hir22b, GKLO22].

Theorem 20 (Symmetry of Information for pK^t under $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$ [GKLO22][20]**).** *If* $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$*, then there exist polynomials p and p_0 such that for all sufficiently large $x, y \in \{0,1\}^*$ and every $t \geq p_0(|x|, |y|)$,*

$$\mathsf{pK}^t(x, y) \geq \mathsf{pK}^{p(t)}(x) + \mathsf{pK}^{p(t)}(y \mid x) - \log p(t).$$

Assuming Theorem 20, we provide a high-level exposition of the proof of a variant of Item 2 from Theorem 19: If $\mathsf{Dist}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{AvgBPP}$ then $\mathsf{NP} \subseteq \mathsf{RTIME}[2^{O(n/\log n)}]$. (A detailed informal presentation of Item 2 of Theorem 19 can be found in [GKLO22, Section 1.3].) Assume that $\mathsf{Dist}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{AvgBPP}$, and let $L \in \mathsf{NP}$. Fix some NP-verifier V for this language. For a string $x \in L$ of length n, let y_x be the lexicographic first string such that $V(x, y_x) = 1$.

1. On the one hand, it follows from Theorem 20 that there is a universal constant $a \geq 1$ such that, for every large enough t, $\mathsf{pK}^{t^a}(y_x \mid x) \leq \mathsf{pK}^t(x, y_x) - \mathsf{pK}^{t^a}(x) + O(\log t)$.

2. On the other hand, under the assumption that $\mathsf{Dist}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{AvgBPP}$, it is possible to prove that, for some universal constant $\varepsilon > 0$ and for every large enough t, $\mathsf{pK}^t(x, y_x) \leq \mathsf{pK}^{t^\varepsilon}(x) + O(\log t)$. This is non-trivial: while it is possible to recover y_x from x with a powerful enough oracle, we must obtain a description of the pair (x, y_x) from (a fixed but arbitrary) x without the aid of such an oracle, using only an average-case easiness assumption.

3. Putting together the previous inequalities from Steps 1 and 2, we get that for every large enough t, $\mathsf{pK}^{t^a}(y_x|x) \leq \mathsf{pK}^{t^\varepsilon}(x) - \mathsf{pK}^{t^a}(x) + O(\log t)$. Consequently, we can upper bound $\mathsf{pK}^{t^a}(y_x|x)$ by the (t^ε, t^a)-probabilistic computational depth of x plus $O(\log t)$, for any $t \geq \text{poly}(n)$, where $n = |x|$.

[20]A more general version of this result is used by [GKLO22] to establish Theorem 19 and its extensions.

4. As in the proof sketch of Theorem 18, one can show that for every x there is some $t(n) = 2^{O(n/\log n)}$ such that $\mathsf{pK}^{t^{\varepsilon}}(x) - \mathsf{pK}^{t^{a}}(x) = O(n/\log n)$. Consequently, using that $\mathsf{pK}^{t_1}(\cdot) \leq \mathsf{pK}^{t_2}(\cdot)$ if $t_1 \geq t_2$, there is a constant $C \geq 1$ such that, for every string x of length n, we have $\mathsf{pK}^{\gamma}(y_x \mid x) \leq C \cdot n/\log n$, where $\gamma(n) = 2^{C \cdot n/\log n}$.

5. Finally, given a positive instance x of L and the upper bound on $\mathsf{pK}^{\gamma}(y_x \mid x)$ from Step 4, we can recover y_x with probability $\geq 2/3$ in time $2^{O(n/\log n)}$. Indeed, this follows from the definition of conditional pK^t complexity: by sampling a random string w of length $2^{C \cdot n/\log n}$ and simulating all machines M of length $\leq C \cdot n/\log n$ on input (x, w) for at most $2^{C \cdot n/\log n}$ steps, we generate y_x with probability at least $2/3$ over the choice of w. Since we can test each string produced in this way using the polynomial-time verifier $V(x, \cdot)$, it follows that $L \in \mathsf{RTIME}[2^{O(n/\log n)}]$.

Remark 21 (Why pK^t complexity?). Both language compression (Theorem 17) and symmetry of information (Theorem 20) are established using techniques from computational pseudorandomness related to the design and analysis of pseudorandom generators (PRGs). This approach has proven extremely useful in time-bounded Kolmogorov complexity (see, e.g., [ABK+06]). In a bit more detail, in the proof of both results we are interested in establishing bounds on the Kolmogorov complexity of a string x. A way of doing this is by considering the string x as a source of "hardness" (e.g., view x as a hard truth-table) in the construction of a generator G^x. The typical analysis of a PRG provides a reconstruction routine, i.e., an algorithm implementing the proof that if we can break G^x using a distinguisher D, then x cannot be hard. In other words, we obtain bounds on the conditional time-bounded Kolmogorov complexity of x given D. Crucially, under assumptions such as $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$, it is often possible to break the corresponding PRG G^x. This provides a powerful way of analysing the time-bounded Kolmogorov complexity of strings in the context of Theorems 18 and 19. More recently, the papers [Hir20b, Hir21] have highlighted the importance of a particular "direct product" generator $\mathsf{G}^x = \mathsf{DP}^x$, which has near-optimal "advice" complexity in its reconstruction procedure and provides tighter bounds on the complexity of x. In the *randomised* reconstruction procedure of DP^x, the advice depends on the particular choice of the random string employed by the procedure, which shows that for a noticeable fraction of random strings w, x has a small description if we are given the random string w. Now observe that this corresponds precisely to pK^t complexity! In previous work [Hir21], this issue is not present because the stronger assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ provides near-optimal derandomisation [BFP05] that allows one to directly get K^t bounds. However, the same PRG is not known to be available under the weaker assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$.

As explained in [GKLO22], while previous works have employed various techniques to *remove* randomness from their arguments in order to analyze K^t

complexity, the idea of *incorporating* randomness in the framework (via pK^t) comes with other benefits beyond the extension of results to the setting of randomised computations. For instance, [CHV22] established *fine-grained* connections between worst-case and average-case complexity. Among other results, they showed that if $\mathsf{NTIME}[n]$ can be deterministically solved in quasi-linear time on average, then $\mathsf{UP} \subseteq \mathsf{DTIME}[2^{O(\sqrt{n \log n})}]$. While the argument from [CHV22] requires the construction of an extremely fast PRG via a delicate analysis, the same result can be proved using pK^t complexity with a simpler proof [GKLO22].

As a potentially accessible direction, we pose the following problem related to Theorem 18 and Item 1 of Theorem 19.

Problem 22. *Show that if* $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$ *then* $\mathsf{NP} \subseteq \mathsf{BPTIME}[2^{O(n/\log n)}]$.

6.3 Learning Algorithms from Probabilistic Average-Case Easiness

This section describes an application of probabilistic Kolmogorov complexity to computational learning theory. More precisely, we show that if $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$, then polynomial-size Boolean circuits can be (agnostically) PAC learned under any samplable distribution in polynomial time. While it is not hard to learn general Boolean circuits under a *worst-case* easiness assumption (e.g, $\mathsf{NP} \subseteq \mathsf{BPP}$) using Occam's razor (see, e.g., [KV94]), here we obtain an interesting consequence for learning under a *weaker* average-case easiness assumption.

The proof adapts a similar learning result from [HN21], established under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ (i.e., average-case easiness for *deterministic* algorithms). This exhibits a natural example of a result that can be lifted to the randomised setting with little effort via pK^t complexity.

Let C be a class of Boolean functions. In the PAC learning model, a learner has access to examples $(x, f(x))$ labelled according to an unknown function $f \in C$. The examples x are drawn according to an unknown probability distribution D_n supported over $\{0,1\}^n$. The goal of the learning algorithm is to produce, with high probability over its internal randomness and draw of labelled examples, a hypothesis h such that $\Pr_{x \sim D_n}[h(x) \neq f(x)] \leq \varepsilon$.

We say that the distribution $D_n \in \mathsf{Samp}[T(n)]/a(n)$ if it can be sampled by an algorithm that runs in time $T(n)$ and has advice complexity $a(n)$. (A sampler described by a uniform machine of code length a counts as advice of length a.) We consider the learnability of the class $C = \mathsf{SIZE}[s]$ of Boolean circuits of size at most $s(n)$, with respect to an unknown distribution D_n from $\mathsf{Samp}[T(n)]/a(n)$.

As in [HN21], the result described below also holds in the more challenging setting of *agnostic learning*, where the function f only needs to be *close* to some function in C. (See [GKLO22] for a concise presentation of this learning model.)

Theorem 23 (Agnostic Learning from Probabilistic Average-Case Easiness of NP [GKLO22])**.**
If $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$*, then for any time constructible functions* $s, T, a \colon \mathbb{N} \to \mathbb{N}$*, and* $\varepsilon \in [0,1]$*,* $\mathsf{SIZE}[s(n)]$ *is agnostic learnable on* $\mathsf{Samp}[T(n)]/a(n)$ *in time* $\mathrm{poly}\big(n, \varepsilon^{-1}, s(n), T(n), a(n)\big)$.

For the proof of this result, the main idea is to design a *random-right-hand-side-refuter* (RRHS-refuter; see [Vad17, KL18]). In short, this is an algorithm that distinguishes the distribution $\big(x^{(1)}, \ldots, x^{(m)}, f(x^{(1)}), \ldots f(x^{(m)})\big)$ from the distribution $\big(x^{(1)}, \ldots, x^{(m)}, b^{(1)}, \ldots b^{(m)}\big)$, where each $x^{(i)}$ is picked from a fixed but unknown distribution D_n, $f \in C$ is a fixed but unknown function and each $b^{(i)}$ is a uniformly random bit. It is known that such an algorithm can be converted into an agnostic learner for C under the distribution D_n.

In [HN21] an efficient RRHS-refuter is constructed using an algorithm that estimates the K^t complexity of a given string, which can be shown to exist under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ [Hir21]. In more detail, [HN21] proved that if a string is sampled from the first distribution, where D_n is efficiently samplable and f is computable by a polynomial size circuit, then it is likely to have bounded K^t complexity (for carefully chosen parameters m and t). On the other hand, using symmetry of information and optimal coding for K^t, which hold under an average-case easiness assumption [Hir21], it can be shown that a random string from the second distribution is likely to have large K^t complexity.

In contrast, under the weaker assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$, we design an efficient algorithm that estimates the pK^t complexity of a given string, which is a more delicate measure than K^t. Combining this algorithm with the symmetry of information for pK^t (Theorem 20), which holds under the same probabilistic average-case easiness assumption, and the optimal coding result for pK^t (Theorem 14), we are able to construct in a similar way an efficient randomised RRHS-refuter. As before, this is sufficient to obtain the desired learning conclusion.

It would be interesting to understand if under the same average-case easiness assumption one can non-trivially learn general Boolean circuits with respect to an arbitrary distribution, i.e., in the standard sense of the PAC learning model.

Problem 24. *Suppose that* $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$*. Is it possible to PAC learn Boolean circuits of size* $O(n)$ (*say, with error* $\varepsilon = 1/10$) *in time* $2^n/n^{\omega(1)}$*?*

We note that this would be possible (via Occam's Razor) if the same average-case easiness assumption led to stronger worst-case upper bounds for languages in NP, such as the conclusion that $\mathsf{NP} \subseteq \mathsf{BPTIME}[2^{n^{0.499}}]$.

7 Probabilistic Versus Deterministic Time-Bounded Kolmogorov Complexity

We have seen that some questions that remain open for classical notions of time-bounded Kolmogorov complexity (such as Kt) can be unconditionally answered in the case of rKt, rK^t, and pK^t. For instance, we presented better bounds for primes with respect to rKt (Theorem 4) and rK^t (Theorem 6) in Section 4, and stated an optimal coding theorem for pK^t (Theorem 14) in Section 5. Moreover, we exhibit several applications of probabilistic time-bounded Kolmogorov complexity to algorithms and complexity in Section 6. It is perhaps a good point to discuss in more detail the relation between deterministic and probabilistic notions of Kolmogorov complexity.

It turns out that, *under strong enough derandomisation hypotheses*, for every string x, its deterministic and probabilistic time-bounded Kolmogorov complexities essentially coincide. For instance, for Kt and rKt we have the following relation.[21]

Theorem 25 ([Oli19])**.** *The following results hold.*

- *If* $\mathsf{promise\text{-}BPE} \subseteq \mathsf{promise\text{-}E}$*, then* $\mathsf{Kt}(x) \leq O(\mathsf{rKt}(x))$ *for every string* x*.*
- *If* $\mathsf{Kt}(x) \leq O(\mathsf{rKt}(x))$ *for every string* x*, then* $\mathsf{BPE} \subseteq \mathsf{E}/O(n)$.

In particular, rKt *and* Kt *are linearly related measures if* $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}\left[2^{\Omega(n)}\right]$.

Note that the connection between derandomisation of probabilistic complexity classes and time-bounded Kolmogorov complexity holds in both directions: in a sense, collapsing Kt and rKt for every string x (up to a constant multiplicative factor) is essentially equivalent to the derandomisation of (promise) BPE, as stated in Theorem 25.

Similarly, under strong enough assumptions, we can show that $\mathsf{pK}^{\mathrm{poly}}(x)$, $\mathsf{rK}^{\mathrm{poly}}(x)$, and $\mathsf{K}^{\mathrm{poly}}(x)$ coincide up to an additive term of order $O(\log n)$.

Theorem 26 ([GKLO22])**.** *The following results hold.*

- *If* $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}\left[2^{\Omega(n)}\right]$*, then there is a polynomial* p *such that* $\mathsf{K}^{p(t)}(x) \leq \mathsf{rK}^t(x) + \log p(t)$*, for every* n*-bit string* x *and time bound* $t(n) \geq n$*.*

[21] Recall that $\mathsf{E} = \mathsf{DTIME}[2^{O(n)}]$ refers to the set of languages that can be decided in deterministic time $2^{O(n)}$, while $\mathsf{BPE} = \mathsf{BTIME}[2^{O(n)}]$ is the set of languages that can be decided in probabilistic time $2^{O(n)}$. The promise version of E is defined in the natural way. Recall that for $\mathsf{promise\text{-}BPE}$ we do not enforce the acceptance probability of the randomised machine to be bounded away from $1/2$ on inputs that do not satisfy the promise.

- *If* $\mathsf{E} \not\subseteq \mathsf{i.o.NSIZE}\left[2^{\Omega(n)}\right]$, *then there is a polynomial* p *such that* $\mathsf{K}^{p(t)}(x) \leq \mathsf{pK}^{t}(x) + \log p(t)$, *for every* n*-bit string* x *and time bound* $t(n) \geq n$.

- *If* $\mathsf{BPE} \not\subseteq \mathsf{i.o.NSIZE}\left[2^{\Omega(n)}\right]$, *then there is a polynomial* p *such that* $\mathsf{rK}^{p(t)}(x) \leq \mathsf{pK}^{t}(x) + \log p(t)$, *for every* n*-bit string* x *and time bound* $t(n) \geq n$.

Proof. We describe the proof of the first item. The other two relations can be established by an appropriate modification of the argument, and we refer to [GKLO22] for the details.

Let $x \in \{0,1\}^n$, and let $t(n) \geq n$. First, the assumption $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}\left[2^{\Omega(n)}\right]$ implies that there is a PRG $G\colon \{0,1\}^{O(\log s)} \to \{0,1\}^s$ that $(1/s)$-fools size-s Boolean circuits and has running time $\mathrm{poly}(s)$ [IW97]. Suppose $\mathsf{rK}^t(x) \leq k$. Let $M \in \{0,1\}^k$ be a probabilistic machine of running time at most t that outputs x with probability at least $2/3$.[22] Consider the following function C on inputs of length t:

$$C(w) = 1 \iff M(w) = x.$$

Clearly, C can be implemented as a $\mathrm{poly}(t)$-size circuit. By definition, the acceptance probability of C is at least $2/3$. Consequently, there is a seed $z \in \{0,1\}^{O(\log t)}$ such that $C(G(z)) = 1$, which in turn implies that $M(G(z)) = x$. This means that, given the description of M and z, we can *deterministically* compute x in time $\mathrm{poly}(t)$. In particular, $\mathsf{K}^{p(t)} \leq k + \log p(t)$, for some large enough polynomial $p(\cdot)$. This polynomial is selected as a function of the overhead in running time and description length caused by the PRG. For this reason, it does not depend on x and t. This completes the proof. $\square$

As a consequence of these (conditional) equivalences, new insights about probabilistic time-bounded Kolmogorov complexity can also shed light on the classical deterministic notions.[23] In particular, if one believes in the corresponding derandomisation assumptions, establishing certain results for rKt, rK^t, and pK^t can be seen as a necessary step before we are able to obtain similar statements for Kt and K^t. One such example is the task of showing better upper bounds on the time-bounded Kolmogorov complexity of prime numbers (Section 4).

Of course, one of the main advantages of probabilistic time-bounded Kolmogorov complexity is that certain results are known unconditionally. In particular, in applications there is often no need to rely on unproven conjectures from complexity theory.

[22] If M runs for more than t steps on some computation path, we simply truncate its computation.

[23] As a concrete example, after proving Theorem 11 in [LO21], we noticed that a similar result also holds for Kt, *unconditionally*. See [LO21] for more information on this.

8 Unconditional Hardness of Estimating Time-Bounded Kolmogorov Complexity

In this section, we turn our attention to *meta-computational* problems, which are problems that are themselves about computations and their complexity. An example of such a problem is MCSP (Minimum Circuit Size Problem), where we are given the truth table of a Boolean function $f\colon \{0,1\}^m \to \{0,1\}$ (represented as a Boolean string x of length $n = 2^m$) and a size bound s, and must decide if f can be computed by a Boolean circuit containing at most s gates. Similarly, we can consider the problem of computing the K^t complexity of an input string $x \in \{0,1\}^n$, where $t(n)$ is some fixed polynomial, such as $t(n) = n^3$. In both cases, it is not hard to see that we obtain a problem in NP. Due to their meta-computational nature, intriguing properties (e.g., [OPS19]), and connections to other areas such as learning theory (e.g., [CIKK16]) and cryptography (e.g., [LP20]), it is possible that the investigation of the complexity of meta-computational problems can offer a fruitful path towards a proof that $\mathsf{P} \neq \mathsf{NP}$.

Given the challenge of establishing strong unconditional lower bounds for problems in NP, it is also interesting to consider the complexity of computing other notions of time-bounded Kolmogorov complexity, such as Kt and rKt. For instance, given a string $x \in \{0,1\}^n$, can we efficiently estimate $\mathsf{Kt}(x)$? Note that this can be done in exponential time using a brute-force search, which places the decision version of this problem in $\mathsf{E} = \mathsf{DTIME}[2^{O(n)}]$. Intuitively, it seems that computing Kt and rKt should be computationally hard for the following reasons:

(*i*) It looks like we must perform an exhaustive search over machines of non-trivial description length.

(*ii*) Thanks to the definitions of Kt and rKt, even the mere act of checking whether a specific machine M prints the string x could require an exponential time simulation.

Note that (*ii*) is not present in problems such as MCSP. (We will revisit this intuition later in the section.)

The next result shows that MrKtP, the Minimum rKt Problem, is computationally hard for randomised algorithms. Indeed, even a gap version of the problem remains difficult. Note that the result provides an *unconditional* complexity lower bound for a natural problem.[24]

[24]As observed by [Oli19], the problem stated next can be solved in randomised exponential time.

Theorem 27 (Complexity Lower Bound for Estimating rKt [Oli19]). *For any* $0 < \varepsilon < 1$, *consider the promise problem* $\Pi^{\varepsilon}_{\mathsf{rKt}} = (\mathcal{YES}_n, \mathcal{NO}_n)_{n \in \mathbb{N}}$, *where*

$$\begin{aligned} \mathcal{YES}_n &= \{x \in \{0,1\}^n \mid \mathsf{rKt} \leq n^{\varepsilon}\}, \\ \mathcal{NO}_n &= \{x \in \{0,1\}^n \mid \mathsf{rKt}(x) \geq n - 1\}. \end{aligned}$$

Then $\Pi^{\varepsilon}_{\mathsf{rKt}} \notin \mathsf{promise\text{-}BPTIME}[n^{\mathrm{polylog}(n)}]$.

Proof Sketch. The proof can be described in different ways. Here we provide a high-level exposition of the argument using insights from computational learning theory. For simplicity, we consider the weaker lower bound $\Pi^{\varepsilon}_{\mathsf{rKt}} \notin \mathsf{promise\text{-}BPP}$.

Assume towards a contradiction that $\Pi^{\varepsilon}_{\mathsf{rKt}} \in \mathsf{promise\text{-}BPP}$. We proceed as follows.

1. Under this assumption, it is possible to show that there is a (promise) *natural property* (in the sense of [RR97]) against functions computed by circuits of size $2^{\delta n}$, for some $\delta > 0$. In other words, we can efficiently distinguish truth-tables of bounded complexity from random truth-tables.

2. By the main result of [CIKK16], this implies that Boolean circuits of size s can be PAC learned under the uniform distribution with membership queries in time $\mathsf{poly}(s)$.

3. Exploring the connection between learning and circuit lower bounds from [OS17a], the existence of such learning algorithms implies that $\mathsf{BPE} \not\subseteq \mathsf{SIZE}(\mathrm{poly})$, where $\mathsf{SIZE}(\mathrm{poly})$ denotes the set of languages computed by Boolean circuits of polynomial size.

4. Finally, we argue that if $\Pi^{\varepsilon}_{\mathsf{rKt}}$ is in $\mathsf{promise\text{-}BPP}$ then $\mathsf{BPE} \subseteq \mathsf{SIZE}(\mathrm{poly})$. Roughly speaking, this step explores techniques from pseudorandomess [ABK+06] to show that every $L \in \mathsf{BPE}$ non-uniformly reduces to $\Pi^{\varepsilon}_{\mathsf{rKt}}$. Since by assumption this problem can be solved by efficient probabilistic algorithms, and such algorithms can be non-uniformly simulated by polynomial-size circuits, the inclusion follows.

Given that Items 3 and 4 are in contradiction, we obtain the desired complexity lower bound. (A proof that employs a different perspective is provided in [Oli19].) $\square$

Curiously, establishing an analogous lower bound for MKtP remains a notorious open problem (see, e.g., [ABK+06]). Here MKtP refers to the problem of deciding, given a string x and a positive integer s, whether $\mathsf{Kt}(x) \leq s$. While it is believed that $\mathsf{MKtP} \notin \mathsf{P}$, we currently only know how to resolve the randomised version of the problem (Theorem 27).[25] This provides another setting where probabilistic time-bounded Kolmogorov complexity offers an advantage over its deterministic counterpart. Note that Theorem 27 implies that $\mathsf{MKtP} \notin \mathsf{BPP}$ under a derandomisation assumption (Theorem 25).

Before presenting a different lower bound, we revise our initial intuition about the hardness of computing rKt and Kt. In light of Corollary 12, a result established after [Oli19], we now understand that the hardness of the gap version of MrKtP can be blamed on Item (*ii*) only. Interestingly, an unexpected algorithmic result sheds light on the hardness of estimating rKt complexity. At the same time, this tells us that different techniques will be needed to understand the computational hardness of problems such as MCSP or computing K^t, where the hardness must come from the analogue of Item (*i*).

Next, we discuss a complexity lower bound for estimating the $\mathsf{rK}^{\mathsf{poly}}$ complexity of an input string.

Theorem 28 (Complexity Lower Bound for Estimating $\mathsf{rK}^{\mathsf{poly}}$ [LOS21])**.** *For any $0 < \varepsilon < 1$ and $d \geq 1$ there exists a constant $k \geq 1$ for which the following holds. Consider the promise problem $\Pi^{\varepsilon,k}_{\mathsf{rK}^t} = (\mathcal{YES}_n, \mathcal{NO}_n)_{n \in \mathbb{N}}$, where*

$$\begin{aligned} \mathcal{YES}_n &= \{x \in \{0,1\}^n \mid \mathsf{rK}^t(x) \leq n^{\varepsilon}\}, \\ \mathcal{NO}_n &= \{x \in \{0,1\}^n \mid \mathsf{rK}^t(x) \geq n - 1\}, \end{aligned}$$

and $t(n) = n^k$. Then $\Pi^{\varepsilon,k}_{\mathsf{rK}^t} \notin \mathsf{promise\text{-}BPTIME}[n^d]$.

Proof. We establish the weaker result that $\Pi^{\varepsilon,k}_{\mathsf{rK}^t} \notin \mathsf{promise\text{-}DTIME}[n^d]$. The lower bound against probabilistic time can be established in a similar way, using that the pseudodeterministic PRG from Theorem 7 also fools probabilistic algorithms (see [LOS21] for the details).

Fix $0 < \varepsilon < 1$ and $d \geq 1$. Let $\varepsilon' = \varepsilon/2$, $d' = 1$, and $c' = d$. Instantiate the pseudodeterministic PRG from Theorem 7 with the parameters ε', c', and d', and assume that $G_n \colon \{0,1\}^{n^{\varepsilon'}} \to \{0,1\}^n$ can be computed probabilistically in time $n^{k'}$, for some constant k' (when provided with the correct advice bit $\alpha'(n)$). We let $k = 2k'$.

Now suppose, towards a contradiction, that $\Pi^{\varepsilon,k}_{\mathsf{rK}^t} \in \mathsf{promise\text{-}DTIME}[n^d]$. Let A be a deterministic algorithm running in time n^d that accepts $\mathcal{YES}_n$ and rejects $\mathcal{NO}_n$, for every large enough n. We argue that the existence of A contradicts the infinitely often guarantee of pseudorandomness provided by the PRG G_n. Indeed, fix a large enough input length n for which G_n succeeds. On the one hand, by our

[25]The proof of Theorem 27 explores randomised computation to perform an indirect diagonalisation, and it is not clear how to implement a similar strategy when only deterministic computations are available.

choice of k and ε', it is easy to see that every string $y \in \{0,1\}^n$ in the image of G_n satisfies $\mathsf{rK}^{n^k}(y) \leq n^{\varepsilon}$. For this reason, $\Pr_{z \sim \{0,1\}^{n^{\varepsilon'}}}[A(G(z)) = 1] = 1$. On the other hand, by a counting argument, a random string $x \sim \{0,1\}^n$ satisfies $\mathsf{rK}^{n^k}(x) \geq n - 1$ with probability $\Omega(1)$ (Proposition 3). This implies that $\Pr_{x \sim \{0,1\}^n}[A(x) = 1] \leq 1 - \Omega(1)$, since A rejects strings in $\mathcal{NO}_n$. Now notice that this violates the pseudorandomness of G_n. In other words, we get that $\Pi^{\varepsilon,k}_{\mathsf{rK}^t} \notin \mathsf{promise\text{-}DTIME}[n^d]$. □

A complexity lower bound for computing K^t against deterministic polynomial-time algorithms and for $t = n^{\omega(1)}$ was established by Hirahara [Hir20b] using different techniques. In both cases, the time bound in the definition of the Kolmogorov complexity measure is larger than the time bound of the algorithm trying to compute or estimate Kolmogorov complexity. Needless to say, it would be extremely interesting to establish a complexity lower bound for computing Kolmogorov complexity with respect to a *fixed* polynomial t in K^t or rK^t that holds against *arbitrary* polynomial-time algorithms (see [LP20]).

A lower bound question that should be more accessible is presented next.

Problem 29 (Exponential Hardness of Estimating rKt)**.** *Show that for any constant* $0 < \varepsilon < 1$ *there is a constant* $\delta > 0$ *such that* $\Pi^{\varepsilon}_{\mathsf{rKt}} \notin \mathsf{promise\text{-}BPTIME}[2^{n^{\delta}}]$.

9 Constructing Strings of Large rKt Complexity and Hierarchy Theorems

The problem of *explicitly* constructing mathematical objects of different types (beyond merely showing their existence) has received much attention in computer science and mathematics. For instance, in Section 1 we described the problem of deterministically producing an n-bit prime. In this section, we are interested in the problem of constructing *incompressible strings*. Some problems of this form are particularly challenging, since given a long incompressible string (e.g., with respect to circuit size or $\mathsf{K}^{\mathrm{poly}}$ complexity), several other constructions problems can be solved (see, e.g., [San12, Kor21]).

In more detail, here we consider the problem of explicitly constructing strings that have large rKt complexity. To provide intuition, let us first consider the much simpler case of Kt complexity. Our goal is to design a deterministic algorithm that, given 1^n, outputs an n-bit string x such that $\mathsf{Kt}(x) \geq n/10$. Does this problem admit a polynomial-time algorithm? It is easy to see that this problem cannot be solved in time $2^{o(n)}$. Indeed, it follows from the very definition of Kt complexity that any deterministic algorithm $A(1^n)$ running in time $2^{o(n)}$ can only print an n-bit string of Kt complexity $o(n)$. However, it is not hard to see that this explicit construction problem can be solved in time $2^{O(n)}$ via an exhaustive search (for instance, by enumerating all strings produced in time $\leq 2^{n/10}$ by machines of description length $\leq n/10$).

Similarly, we ask if there is an algorithm that runs in time $2^{O(n)}$ and produces an n-bit string x such that $\mathsf{rKt}(x) \geq n/10$. The natural brute-force approach to solve this problems involves the simulation of *randomised* algorithms. For this reason, we relax our goal as follows: Is there a *randomised* algorithm $A(1^n)$ that runs in time $2^{O(n)}$ and outputs with probability at least $2/3$ a *fixed* n-bit string w_n such that $\mathsf{rKt}(w_n) \geq n/10$? In other words, we would like to have a *pseudodeterministic* construction of strings of large rKt complexity, in the sense of [GG11].

A careful inspection of the natural brute-force approach that works for Kt reveals that it simply does not work in the case of rKt: roughly speaking, the simulation of different randomised machines comes with uncertainties, and it is not clear if after all the simulations we isolate the same string w_n with high probability.

In [LOS21], we connected the problem of constructing strings of large rKt complexity to the longstanding question of establishing a strong time hierarchy theorem for probabilistic computations. Recall that, while it is known that $\mathsf{BPEXP} \not\subseteq \mathsf{BPP}$, it is consistent with current knowledge that inclusions such as $\mathsf{BPTIME}[2^n] \subseteq \mathsf{BPTIME}[2^{n^{0.01}}]$ and $\mathsf{BPTIME}[n^{50}] \subseteq \mathsf{BPTIME}[n^2]$ might hold.[26]

Theorem 30 (Explicit Construction Problem for rKt and Probabilistic Time Hierarchies)**.** *The following statements are equivalent:*

(1) Pseudodeterministic construction of strings of large rKt complexity: *There is a constant $\varepsilon > 0$ and a randomised algorithm A that, given m, runs in time $2^{O(m)}$ and outputs with probability at least $2/3$ a fixed m-bit string w_m such that $\mathsf{rKt}(w_m) \geq \varepsilon m$.*

(2) Strong time hierarchy theorem for probabilistic computation: *There are constants $k \geq 1$ and $\lambda > 0$ for which the following holds. For any constructive function $n \leq t(n) \leq 2^{\lambda \cdot 2^n}$, there is a language $L \in \mathsf{BPTIME}[(t(n)^k]$ such that $L \notin \mathsf{i.o.BPTIME}[t(n)]/\log t(n)$.*

The proof of Theorem 30 is elementary, and proceeds by associating with a language L a sequence of truth-tables, one for each input length n (each truth-table can be seen as a string of length $m = 2^n$). For a sketch of the argument and a detailed proof, see [LOS21].

Note that the connection between the explicit (pseudodeterministic) construction problem for rKt and hierarchy theorems goes in both ways. More generally, [LOS21] explored the fruitful relation between pseudodeterministic PRGs (see

[26]Some separations have been established if we allow advice bits in the upper bound and lower bound. For instance, $\mathsf{BPTIME}[n^{50}]/1 \not\subseteq \mathsf{BPTIME}[n^2]/1$ (see, e.g., [Bar02, FS04]).

Section 4), the explicit construction problem for rKt complexity, and hierarchy theorems for probabilistic time to make advances in all these areas. On the other hand, [LO21] connected rKt complexity and its coding theorem (Theorem 10) to the study of time hierarchy theorems for sampling distributions (cf., [Wat14]).

10 Concluding Remarks

We presented key results in probabilistic Kolmogorov complexity and applications to several areas, including explicit constructions, complexity lower bounds, sampling algorithms, average-case complexity, and learning theory. The probabilistic measures rK^t, pK^t, and rKt are particularly useful in settings that involve randomised algorithms. While it is quite possible for these complexity measures to be essentially equivalent to their deterministic counterparts (Section 7), they allow us to obtain unconditional results that do not rely on derandomisation assumptions. In some cases, probabilistic Kolmogorov complexity can significantly simplify existing arguments or is the only known approach to certain results.

The results presented in the preceding sections naturally suggest several problems and directions. For example, we believe that it should be possible to make progress on the following fronts:

– Designing improved pseudodeterministic PRGs and obtaining better upper bounds on the rKt complexity of prime numbers.

– Establishing new unconditional lower bounds on the complexity of meta-computational problems such as MKtP and MrKtP.

For a more precise formulation of these problems, we refer to the concrete questions stated in the corresponding sections of the article (Section 4 and Section 8). Additional questions of interest are presented in other parts of the survey.

Given the number of recent advances and applications of time-bounded Kolmogorov complexity to algorithms and complexity theory (see Section 1), it is hard to predict which directions will be more fruitful. Nevertheless, we are particularly optimistic about the role that probabilistic Kolmogorov complexity can take in the investigation of the relations between average-case complexity and worst-case complexity, cryptography, and learning algorithms. In particular, analogously to results of [Hir21], under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$, all main pillars of Kolmogorov complexity are known to hold for pK^t complexity: incompressibility (Proposition 3), coding theorem (Theorem 14), language compression (Theorem 17), and symmetry of information (Theorem 20). Taking into account the wide applicability of these results and the ubiquitous role of randomised algorithms in theoretical computer science, we expect to see further developments in average-case complexity powered by tools and perspectives from probabilistic Kolmogorov complexity.

References

[Aar14] Scott Aaronson. The equivalence of sampling and searching. *Theory Comput. Syst.*, 55(2):281–298, 2014.

[ABK+06] Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006.

[AF09] Luis Antunes and Lance Fortnow. Worst-case running times for average-case algorithms. In *Conference on Computational Complexity* (CCC), pages 298–303, 2009.

[AFvM01] Luis Antunes, Lance Fortnow, and Dieter van Melkebeek. Computational depth. In *Conference on Computational Complexity* (CCC), pages 266–273, 2001.

[AKS02] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Ann. of Math.*, 2:781–793, 2002.

[All92] Eric Allender. Applications of time-bounded Kolmogorov complexity in complexity theory. In *Kolmogorov complexity and computational complexity*, pages 4–22. Springer, 1992.

[All01] Eric Allender. When worlds collide: Derandomization, lower bounds, and Kolmogorov complexity. In *International Conference on Foundations of Software Technology and Theoretical Computer Science* (FSTTCS), pages 1–15. Springer, 2001.

[All17] Eric Allender. The complexity of complexity. In *Computability and Complexity*, pages 79–94. Springer, 2017.

[All21] Eric Allender. Vaughan Jones, Kolmogorov complexity, and the new complexity landscape around circuit minimization. *New Zealand Journal of Mathematics*, 52:585–604, 2021.

[Bar02] Boaz Barak. A probabilistic-time hierarchy theorem for “slightly non-uniform" algorithms. In *International Workshop on Randomization and Approximation Techniques* (RANDOM), pages 194–208, 2002.

[BFP05] Harry Buhrman, Lance Fortnow, and Aduri Pavan. Some results on derandomization. *Theory Comput. Syst.*, 38(2):211–227, 2005.

[BLvM05] Harry Buhrman, Troy Lee, and Dieter van Melkebeek. Language compression and pseudorandom generators. *Comput. Complex.*, 14(3):228–255, 2005.

[BMVZ18] Bruno Bauwens, Anton Makhlin, Nikolai K. Vereshchagin, and Marius Zimand. Short lists with short programs in short time. *Comput. Complex.*, 27(1):31–61, 2018.

[BT06] Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Found. Trends Theor. Comput. Sci.*, 2(1), 2006.

[CHV22] Lijie Chen, Shuichi Hirahara, and Neekon Vafa. Average-case hardness of NP and PH from worst-case fine-grained assumptions. In *Innovations in Theoretical Computer Science* (ITCS), 2022.

[CIKK16] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In *Conference on Computational Complexity* (CCC), pages 10:1–10:24, 2016.

[For04] Lance Fortnow. Kolmogorov complexity and computational complexity. *Complexity of Computations and Proofs. Quaderni di Matematica*, 13, 2004.

[FS04] Lance Fortnow and Rahul Santhanam. Hierarchy theorems for probabilistic polynomial time. In *Symposium on Foundations of Computer Science* (FOCS, pages 316–324, 2004.

[GG11] Eran Gat and Shafi Goldwasser. Probabilistic search algorithms with unique answers and their cryptographic applications. *Electronic Colloquium on Computational Complexity* (ECCC), 18:136, 2011.

[GK22] Halley Goldberg and Valentine Kabanets. A simpler proof of the worst-case to average-case reduction for polynomial hierarchy via symmetry of information. *Electron. Colloquium Comput. Complex.*, 7:1–14, 2022.

[GKLO22] Halley Goldberg, Valentine Kabanets, Zhenjian Lu, and Igor C. Oliveira. Probabilistic Kolmogorov complexity with applications to average-case complexity. In *Computational Complexity Conference* (CCC), 2022.

[Hir18] Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *Symposium on Foundations of Computer Science* (FOCS), pages 247–258, 2018.

[Hir20a] Shuichi Hirahara. Characterizing average-case complexity of PH by worst-case meta-complexity. In *Symposium on Foundations of Computer Science* (FOCS), pages 50–60, 2020.

[Hir20b] Shuichi Hirahara. Unexpected hardness results for Kolmogorov complexity under uniform reductions. In *Symposium on Theory of Computing* (STOC), pages 1038–1051, 2020.

[Hir21] Shuichi Hirahara. Average-case hardness of NP from exponential worst-case hardness assumptions. In *Symposium on Theory of Computing* (STOC), pages 292–302, 2021.

[Hir22a] Shuichi Hirahara. Meta-computational average-case complexity: A new paradigm toward excluding heuristica. *Bull. EATCS*, 136, 2022.

[Hir22b] Shuichi Hirahara. Symmetry of information in heuristica. Manuscript, 2022.

[HN21] Shuichi Hirahara and Mikito Nanashima. On worst-case learning in relativized heuristica. In *Symposium on Foundations of Computer Science* (FOCS), 2021.

[HS22] Shuichi Hirahara and Rahul Santhanam. Errorless versus error-prone average-case complexity. In *Innovations in Theoretical Computer Science Conference* (ITCS), 2022.

[Ila20] Rahul Ilango. Connecting Perebor conjectures: Towards a search to decision reduction for minimizing formulas. In *Computational Complexity Conference* (CCC), 2020.

[Ila21] Rahul Ilango. The minimum formula size problem is (ETH) hard. In *Symposium on Foundations of Computer Science* (FOCS), pages 427–432, 2021.

[ILO20] Rahul Ilango, Bruno Loff, and Igor C. Oliveira. NP-hardness of circuit minimization for multi-output functions. In *Computational Complexity Conference* (CCC), 2020.

[IP01] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

[IRS21] Rahul Ilango, Hanlin Ren, and Rahul Santhanam. Hardness on any samplable distribution suffices: New characterizations of one-way functions by meta-complexity. *Electron. Colloquium Comput. Complex.*, page 82, 2021.

[IW97] Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Symposium on Theory of Computing* (STOC), pages 220–229. ACM, 1997.

[KL18] Pravesh K. Kothari and Roi Livni. Improper learning by refuting. In *Innovations in Theoretical Computer Science Conference* (ITCS), pages 55:1–55:10, 2018.

[Ko91] Ker-I Ko. On the complexity of learning minimum time-bounded Turing machines. *SIAM J. Comput.*, 20(5):962–986, 1991.

[Kor21] Oliver Korten. The hardest explicit construction. In *Symposium on Foundations of Computer Science* (FOCS), pages 433–444, 2021.

[Kra22] Jan Krajíček. Information in propositional proofs and algorithmic proof search. *The Journal of Symbolic Logic*, 2022.

[KV94] Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[Lee06] Troy Lee. *Kolmogorov complexity and formula lower bounds*. PhD thesis, University of Amsterdam, 2006.

[Lev84] Leonid A. Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61(1):15–37, 1984.

[LO87] Jeffrey C. Lagarias and Andrew M. Odlyzko. Computing π(x): An analytic method. *J. Algorithms*, 8(2):173–191, 1987.

[LO21] Zhenjian Lu and Igor C. Oliveira. An efficient coding theorem via probabilistic representations and its applications. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 94:1–94:20, 2021.

[LOS21] Zhenjian Lu, Igor C. Oliveira, and Rahul Santhanam. Pseudodeterministic algorithms and the structure of probabilistic time. In *Symposium on Theory of Computing* (STOC), pages 303–316, 2021.

[LOZ22] Zhenjian Lu, Igor C. Oliveira, and Marius Zimand. Optimal coding theorems in time-bounded Kolmogorov complexity. In *International Colloquium on Automata, Languages, and Programming* (ICALP), 2022.

[LP20] Yanyi Liu and Rafael Pass. On one-way functions and Kolmogorov complexity. In *Symposium on Foundations of Computer Science* (FOCS), pages 1243–1254, 2020.

[LP21] Yanyi Liu and Rafael Pass. On the possibility of basing cryptography on EXP≠BPP. In *Annual International Cryptology Conference* (CRYPTO), pages 11–40, 2021.

[LV92] Ming Li and Paul M. B. Vitányi. Average-case complexity under the universal distribution equals worst-case complexity. *Inf. Process. Lett.*, 42(3):145–149, 1992.

[LV19] Ming Li and Paul M. B. Vitányi. *An introduction to Kolmogorov complexity and its applications*. Springer, 2019.

[Oli19] Igor C. Oliveira. Randomness and intractability in Kolmogorov complexity. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 32:1–32:14, 2019.

[OPS19] Igor C. Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. In *Computational Complexity Conference* (CCC), pages 27:1–27:29, 2019.

[OS17a] Igor C. Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *Computational Complexity Conference* (CCC), pages 18:1–18:49, 2017.

[OS17b] Igor C. Oliveira and Rahul Santhanam. Pseudodeterministic constructions in subexponential time. In *Symposium on Theory of Computing* (STOC), pages 665–677, 2017.

[RR97] Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.

[RS21] Hanlin Ren and Rahul Santhanam. Hardness of KT characterizes parallel cryptography. In *Computational Complexity Conference* (CCC), pages 35:1–35:58, 2021.

[RSZ21] Andrei E. Romashchenko, Alexander Shen, and Marius Zimand. 27 open problems in Kolmogorov complexity. *SIGACT News*, 52(4):31–54, 2021.

[San12] Rahul Santhanam. The complexity of explicit constructions. *Theory Comput. Syst.*, 51(3):297–312, 2012.

[Sip83] Michael Sipser. A complexity theoretic approach to randomness. In *Symposium on Theory of Computing* (STOC), pages 330–335, 1983.

[SUV17] Alexander Shen, Vladimir Andreyevich Uspensky, and Nikolay Vereshchagin. *Kolmogorov Complexity and Algorithmic Randomness*. American Mathematical Society, 2017.

[TCH12] Terence Tao, Ernest Croot, III, and Harald Helfgott. Deterministic methods to find primes. *Math. Comp.*, 81(278):1233–1246, 2012.

[Vad17] Salil P. Vadhan. On learning vs. refutation. In *Conference on Learning Theory* (COLT), 2017.

[Wat14] Thomas Watson. Time hierarchies for sampling distributions. *SIAM J. Comput.*, 43(5):1709–1727, 2014.