Kolmogorov Complexity: An Algorithmic Theory of Information

Merkrouris Papamichail, Giorgos Roussakis
merkourisp@csd.uoc.gr, roussakis@csd.uoc.gr

Computer Science Department
University of Crete

Fall 2023

Abstract

In this paper we present some elementary results for the literature regarding Kolmogorov complexity. The notion of Kolomogorov complexity measures the complexity of an object by the length of the sorter program that prints that object and then halts. This provides an algorithmic view of information. In this paper we firstly describe the related concepts intuitively. Then present some formal definitions, and basic results. We conclude by considering the relation between Kolmogorov complexity and another popular notion of information, namely Shannon's entropy.

1 Introduction

Kolmogorov complexity, a concept rooted in algorithmic information theory, serves as a powerful tool for measuring the inherent complexity of a finite object or sequence. Developed by Andrey Kolmogorov in the 1960s, this notion revolves around the idea of describing an object through the shortest possible algorithm that generates it. In essence, Kolmogorov complexity quantifies the information content of an object by evaluating the length of the most concise program that produces it. This elegant approach to complexity has applications ranging from data compression to the understanding of randomness, playing a fundamental role in the foundations of computer science and information theory.

We begin by giving an intuitive example. Consider the two following sequences of natural numbers.

$$s_1 : 2, 3, 5, 7, 11, 13, 17, 19, 23, \ldots$$
$$s_2 : 3, 5, 17, 257, 65537, 4294967297, 18446744073709551617, \ldots$$

How can we measure the *complexity* or *information* of each sequence? We could consider the popular Shannon's information theory (or just information theory). In this formalism the information contained to an object is measured by the unexpectedness that this object will be observed, with respect to a probability distribution p. Thus, a sender could toss a fair coin to decide which of the sequences s_1, s_2 to send. In this case, both messages can be regarded to contain the same amount of information, only depended by the probability p. On the other hand, we can argue that the sequence s_1 carries more information than s_2. The sequence s_1 enumerates the *prime numbers*, while s_2 is the sequence of the Fermat numbers. Recall that there is no closed formula that gives the sequence of the prime numbers, while the sequence of the Fermat numbers can be generated by the closed formula,

$$F_n = 2^{2^n} + 1. \tag{1}$$

Essentially, the infinite sequence s_2 is compressible to (1), since a receiver can reconstruct s_2 only by knowing (1). Contrarily, s_1 is *not* in the same manner.

This duality between the size of the object to be described and the length of the a minimum programme that generates this object is at the core of computer science. For instance in the early days of the "personal computer" music was encoded as notes of particular instruments (see MIDI protocol), essentially a programme that generated the desired sound. This technique considerably reduces the sound's size, compared to storing the sound as samples. Other practical applications can be found in image and video processing (see Figure 1).

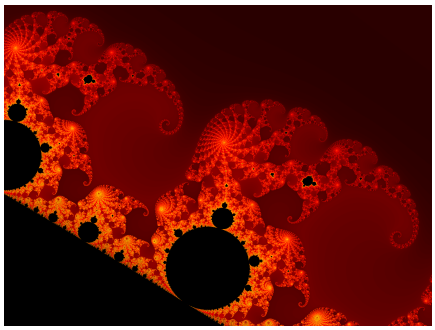

Figure 1: A part of the Mandelbrot set fractal. Simply storing the 24-bit color of each pixel in this image would require 23 million bytes, but a small computer programme can reproduce these 23MB using the definition of Mandelbort set and the corner coordinates of the image, namely a few dozen bytes in any programming language.

The relationship between an object and its description is captured by the notion of Kolmogorov complexity. Despite this notion is not restricted to finite objects, or the format of objects examined; here we will consider only *finite binary sequences*.

2 Basic Definitions

In this section we present our basic definitions and results. An important issue that needs to be addressed before we give a formal definition of Kolomogorov complexity is the *description language*. In other words, which language we will use to describe each object. Since we want our description language to describe a computer programme, we could use any programming language, such as C, Prolog or Java. On the other hand, we could use an abstract formalism such as the *Turing machine* (TM). Any algorithm can be represented by a Turing machine. Moreover, any TM can be *simulated* by the *universal Turing machine* $\mathcal{U}$, which essentially consists an abstract model for any general purpose computer. The universal TM $\mathcal{U}$ accepts an encoding $\langle M \rangle$ of any TM M as a *finite binary sequence*, simulates M and halts. We give the following formal definition.

Definition 2.1 (Kolmogorov Complexity)**.** *Let $s \in \{0,1\}^\star$ be a finite sequence. The* Kolmogorov *complexity of s is the minimum, in length, TM M, such that M prints s and halts. Namely,*

$$K(s) = \min\{|\langle M \rangle| \mid \mathcal{U}(\langle M \rangle) = s\}. \tag{2}$$

When we want to highlight the fact that the Kolmogorov complexity is measured with respect to the universal Turing machine $\mathcal{U}$, we will write $K_{\mathcal{U}}(s)$. Naturally, for any string $s \in \{0,1\}^\star$ there is the *trivial* TM that just prints s bit-by-bit and then stops. We denote this trivial machine by P_s. Note that such a machine will have internally "hard-coded" the string s. Therefore,

$$K(s) \leq |\langle P_s \rangle| = |s| + c, \tag{3}$$

where c is a constant depending only in the computational model.

In the following theorem we argue that Kolmogorv complexity is inherited to the string s and independent of the choice of the formalism, e.g. $\mathcal{U}$. A fundamental doctrine of computer science is the *Turing-Church thesis*, which states that *any reasonable computational model is equivalent to the universal Turing machine $\mathcal{U}$*. Namely, for any other reasonable computational model $\mathcal{A}$, there is a computable mapping $\phi_{\mathcal{U}\to\mathcal{A}}(\cdot)$, such that $\mathcal{U}(\langle M \rangle) = \mathcal{A}(\phi_{\mathcal{U}\to\mathcal{A}}(\langle M \rangle))$. Intuitively, $\phi_{\mathcal{U}\to\mathcal{A}}(\cdot)$ is a *compiler* from the programming language $\mathcal{U}$ to the programming language $\mathcal{A}$. The Church-Turing thesis states that such a compiler exists for any two reasonable computational models.

Theorem 2.1 (Universality)**.** *Let $s \in \{0,1\}^\star$ be a finite binary string. If $\mathcal{U}$ is a universal Turing machine, for any other complete, deterministic computational model $\mathcal{A}$ there is a constant $c_{\mathcal{A}}$ such that*

$$K_{\mathcal{U}}(s) \leq K_{\mathcal{A}}(s) + c_{\mathcal{A}\to\mathcal{U}} \tag{4}$$

We describe the intuition behind the equation (4). Assume that we want to calculate the Kolmogorov complexity in the programming language $\mathcal{U}$. A description of s in $\mathcal{U}$ can be derived by taking a description of s in the programming language $\mathcal{A}$ and simulating it in $\mathcal{U}$. Thus, the size of the minimum description of s in $\mathcal{U}$ is bounded by the minimum description of s in $\mathcal{A}$, plus the size of an $\mathcal{A}$ to $\mathcal{U}$ compiler. The latter term is denoted by the constant $c_{\mathcal{A}\to\mathcal{U}}$, namely $|\phi_{\mathcal{A}\to\mathcal{U}}(\cdot)|$. The constant $c_{\mathcal{A}\to\mathcal{U}}$ in equation (4) may be very large. For example, $\mathcal{A}$ may be a high level programming language with a large number of built-in commands. The language $\mathcal{U}$ can be low-level assembly. The compiler program will contain the details of the implementation of all these functions. The crucial point is that the length of this compiler is *independent* of the length of s, the string to be compressed. For sufficiently long s, the length of this compiler can be neglected.

3 Elementary Results

In this Section we present some elemental results from the literature. We begin from two negative results that limit our expectations regarding both the ideal amount of compression that can be achieved, and the inability to reach that ideal compression.

3.1 Incompressible Strings

We give a simple combinatorial argument to bound the size of the set of compressible strings. Assume the set $S^n = \{0,1\}^n$ of binary string with length at most n. Naturally, we have $|S^n| = 2^n$. On the other hand, consider the set M^k of TMs with length at most k, namely $M^k = \{M \mid |\langle M\rangle| \leq k\}$. Symmetrically, we have $|M^k| \leq 2^k$, since not every binary string of length at most k will represent a valid description of a TM. Since every TM $M \in M^k$ will generate at most one string in S^n. Only 2^k strings in S^n can be compressed with a description of length shorter than k. Therefore, the size of incompressible strings is bounded bellow by $2^n - 2^k$. Assuming a uniform distribution on S^n the probability of choosing a compressible string is $\frac{2^k}{2^n} = 2^{k-n}$ (see Figure 2). We formally present our observation in the following theorem.

Theorem 3.1 (Incompressible Strings)**.** *Let $S^n = \{0,1\}^n$ be the set of strings with length at most n. For every $k \in \mathbb{N}$, there are at least $2^n - 2^k$ strings with $K(s) > k$.*

This is anything but unimportant, as it tells us that most sequences have a complexity close to their length. As an example, the fraction of sequences of length n that have complexity less than n-5 is $\frac{1}{32}$. Obviously, this is a cornerstone of algorithmic information theory, but has also seen wide applicability in algorithmic probability and inductive inference, fields related to Kolmogorov and Solomonoff, data compression as it sets practical limitations in the development of compression algorithms and data encryption as pseudorandom sequences generated by algorithms can be compressed, but truly random sequences provide a level of unpredictability

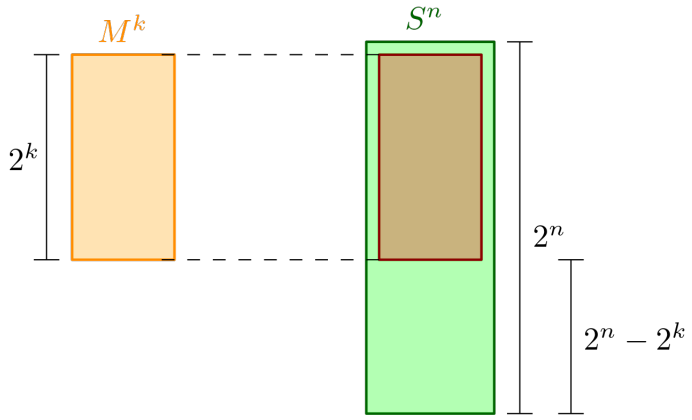

Figure 2: The size of the set of incompressible strings, relative to the size of the set $\{0,1\}^n$. Since any TM considered is deterministic, we may consider an *injection* from TMs with descriptions of length at most k to the set of strings of length at most n.

that is valuable in encryption. Some major results that have arisen from the utilization of the aforementioned theorem and theory, are the Shellsort average case lower bound[3], the average case analysis for heapsort[6], in Turing machines and Automata the proof that two heads are better than two tapes [4]

3.2 Non-computability

For any computational model, there is no general algorithm to decide whether a program will halt or not. This is known as the *halting problem* and is a fundamental fact of computer science. Obviously there are many programs that can be shown to halt or that go on forever. The halting problem states that it *does not* exist a general algorithm that answers this question for any programme. This shares a similar intuition and importance with one of the greatest achievements of 20th mathematics, Gödel's incompletness theorem. In 1931, Gödel used the idea of self-reference to show that any interesting system of mathematics is not complete; there are statements in the system that are true but that cannot be proved within the system.

A consequence of the halting problem is the non-computability of Kolmogorov complexity. The only way to find the shortest program in general is to try all short programs and see which of them can do the job. However, at any time some of the short programs may not have halted and there is no effective (finite mechanical) way to tell whether or not they will halt and what they will print out. Hence, there is no automatic way to find the shortest program to print a given string.

Theorem 3.2 (Non-computability of Kolmogorov Complexity)**.** *The function* $K\colon \{0,1\}^\star \to \mathbb{N}$ *that gives the Kolmogorov complexity of any finite binary string* $s \in \{0,1\}^\star$ *is* not

computable.

3.3 Universal Probability

As we established earlier, the most sequences of length n have complexity close to n. Since the probability of an input program p is $2^{-l(p)}$, shorter programs are much more probable than longer ones, and when they produce long strings, shorter programs do not produce random strings, they produce strings with simply described structure.

Definition 3.1. *The universal probability of a string x is*

$$P_{\mathcal{U}}(x) = \sum_{p:\mathcal{U}(p)=x} 2^{-l(p)} = Pr(\mathcal{U}(p) = x)$$

In other words this is the probability that a program randomly drawn as a sequence of fair coin flips $p_1, p_2, \ldots$ will print out the string x.

The following theorem relates the universal probability $P_{\mathcal{U}}(\cdot)$ with Kolmogorov complexity.

Theorem 3.3. *There exists a constant c, independent of x, such that:*

$$2^{-K(x)} \leq P_{\mathcal{U}}(x) \leq c2^{-K(x)}$$

for all strings x. Thus, the universal probability of a string x is determined essentially by its Kolmogorov complexity.

This constitutes a major result in the field as it implies that K(x) and $\log \frac{1}{P_{\mathcal{U}(x)}}$ have equal status as universal complexity measures.

3.4 The Ω number

In this subsection we will talk about Chaitin's mystical, magical number, Ω, which has some interesting properties:

Definition 3.2 (The Ω number [1])**.** *The Chaitin's Ω number is defined as the probability of a randomly chosen programme p halts. Namely,*

$$\Omega = \sum_{p \ s.t. \ \mathcal{U}(p) \ halts} 2^{-l(p)}.$$

The properties of Ω are:

1. Ω is incomputable. From the definition of the halting problem we can see that there is no effective way to compute Ω.

2. Ω is a "philosopher's stone". Knowing Ω to an accuracy of n bits will enable us to decide the truth of any provable or finitely refutable mathematical theorem that can be written in less than n bits. The basic idea of the procedure using n bits of Ω is simple: We run all programs until the sum of the masses $2 - l(p)$ contributed by programs that halt equals or exceeds $\Omega_n = 0.\omega_1\omega_2 \ldots \omega_n$, the truncated version of Ω that we are given. Then since:

$$\Omega - \Omega_n < 2^{-n}$$

What are some of the questions that we can resolve using Ω? Many of the interesting problems in number theory can be stated as a search for a counterexample. For example, it is straightforward to write a program that searches over the integers x, y, z, and n and halts only if it finds a counterexample to Fermat's last theorem, which states that

$$x^n + y^n = z^n$$

has no solution in integers $n \geq 3$. Another example is Goldbach's conjecture, which states that any even number is a sum of two primes. Our program would search through all the even numbers starting with 2, check all prime numbers less than it and find a decomposition as a sum of two primes. It will halt if it comes across an even number that does not have such a decomposition. Knowing whether this program halts is equivalent to knowing the truth of Goldbach's conjecture.

3. Ω is algorithmically random

3.5 Minimum Description Length

Suppose we have $X_1, X_2, \ldots, X_n$ be drawn i.i.d. from probability mass function p(x). Assuming we don't know p(x) but know $p(x) \in \mathcal{P}$, a class of probability mass functions. We want to estimate a PMF that best fits the data. For simple classes P, we do the maximum likelihood procedure. However, if P is rich enough there is a problem of overfitting the data. Various methods have been applied to deal with this problem. In the simplest case we assume that it comes from some parametric distribution and the parameters of the distribution are estimated from the data. A more general procedure is to take the maximum likelihood estimate and smooth it out to obtain a smooth density. With enough data, and appropriate smoothness conditions, it is possible to make good estimates of the original density. This process is called kernel density estimation.
However, the theory of Kolmogorov complexity (or the Kolmogorov sufficient statistic) suggests a different procedure: Find the $p \in \mathcal{P}$ that minimizes:

$$L_P(X_1, X_2, \ldots, X_n) = K(p) + log\frac{1}{p(X_1, X_2, \ldots, X_n)}$$

In words, the Minimum Description Length(MDL) principle: Given the data and choice of models, choose the model such that the description of the model plus the conditional description of the data is as short as possible.

This is an initial formulation of Minimum Description Length which is sometimes referred to as idealized MDL[2]. This p is known as Kolmogorov minimum sufficient statistic. While this procedure is mathematically well defined, it cannot be used in practice. The reason is that in general, the P minimizing the previous formula, cannot be effectively computed. Let's look at another more practical formulation:
Suppose we are given data D and a model/theory, a hypothesis H. Another definition which is more widely applicable is the following. Select a hypothesis H that minimizes:

$$K(H) + K(D|H)$$

which is the code-independent, computably invariant, absolute form of the MDL principle.
In other words, the goal of statistical inference may be cast as trying to find regularity in the data. 'Regularity' may be identified with 'ability to compress'. MDL combines these two insights by viewing learning as data compression: it tells us that, for a given set of hypotheses H and data set D, we should try to find the hypothesis or combination of hypotheses in H that compresses D most.

4 Relation to Shannon's Entropy

Suppose we have source words x distributed as a random variable X with probability P(x). While K(x) is fixed for each x and gives the shortest code word length(only up to a fixed constant) and is independent of the probability distribution P, we may wonder whether K is also universal in the following sense: If we weigh each individual code word length for x with its probability P (x) the resulting P expected code word length $\sum_x P(x)K(x)$ achieves the minimal average code word length $H(P) = -\sum_x P(x) \log P(x)$? Here we sum over the entire support of P; restricting summation to a small set, for example the singleton set $\{x\}$, can give a different result. The reasoning above implies that, under some mild restrictions on the distributions P , the answer is yes.

This is expressed in the following theorem, where, instead of the quotient we look at the difference of $\sum_x P(x)K(x)$ and $H(P)$. This allows us to express really small distinctions. We call an information source P recursive if there exists a Turing machine that, computes it. The following theorem can be found in Li and Vitanyi(1997)[5]:

Theorem 4.1. *Let P be a recursive information source. Then for all n,*

$$0 \leq \sum_{x \in \{0,1\}^n} P^(x)K(x) - H(P^) \leq c_P$$

where c_P is a constant that depends only on P(and not on n).

Therefore, for every computable distribution P, the universal code D^* whose length function is the Kolmogorov complexity compresses on average at least as much as the Shannon–Fano code for P . This is the intuitive reason why, no matter what computable distribution P we take, its expected Kolmogorov complexity is close to its entropy.

5 Conclusions

As we embarked on this project, we started with the prospect of understanding a concept that seemed interesting. However, as we delved deeper into the theory and its implications we found ourselves almost overwhelmed by the breadth and depth of the field of Algorithmic Information Theory. The starting point was Kolmogorov complexity, a perhaps simple notion of determining the complexity of a string by the complexity of the program producing it. The concepts we studied in the process form a rich tapestry that transcends the boundaries of computation, information theory, and artificial intelligence.
The fundamental premise of Kolmogorov complexity, measuring the inherent complexity of finite objects through the shortest algorithm, has laid the groundwork for an exploration into the nature of information. The exploration of incompressibility has underscored the significance of certain data resisting reduction, contributing to our understanding of data compression and the nature of information representation.

References

[1] Thomas M Cover and Joy A Thomas. "Network information theory". In: *Elements of information theory* (1991), pp. 374–458.

[2] Peter D Grünwald. *The minimum description length principle*. MIT press, 2007.

[3] Tao Jiang, Ming Li, and Paul Vitányi. "A lower bound on the average-case complexity of Shellsort". In: *Journal of the ACM (JACM)* 47.5 (2000), pp. 905–911.

[4] Tao Jiang, Joel I Seiferas, and Paul MB Vitányi. "Two heads are better than two tapes". In: *Journal of the ACM (JACM)* 44.2 (1997), pp. 237–256.

[5] Ming Li, Paul Vitányi, et al. *An introduction to Kolmogorov complexity and its applications*. Vol. 3. Springer, 2008.

[6] Russel Schaffer and Robert Sedgewick. "The analysis of heapsort". In: *Journal of Algorithms* 15.1 (1993), pp. 76–100.