entropy

MDPI

Review

How Incomputable Is Kolmogorov Complexity?

Paul M.B. Vitányi [1,2]

[1] The National Research Center for Mathematics and Computer Science in the Netherlands (CWI), 1098XG Amsterdam, The Netherlands; paulv@cwi.nl

[2] Department of Computer Science, University of Amsterdam, 1012 WX Amsterdam, The Netherlands

Received: 19 February 2020; Accepted: 29 March 2020; Published: 3 April 2020

Abstract: Kolmogorov complexity is the length of the ultimately compressed version of a file (i.e., anything which can be put in a computer). Formally, it is the length of a shortest program from which the file can be reconstructed. We discuss the incomputability of Kolmogorov complexity, which formal loopholes this leaves us with, recent approaches to compute or approximate Kolmogorov complexity, which approaches are problematic, and which approaches are viable.

Keywords: Kolmogorov complexity; incomputability; feasibility

1. Introduction

Recently there have been several proposals regarding how to compute or approximate in some fashion the Kolmogorov complexity function. There is a proposal that is popular as a reference in papers that do not care about theoretical niceties, and a couple of proposals that do make sense but are not readily applicable. Therefore, it is timely to survey the field and show what is and what is not proven.

The plain Kolmogorov complexity was defined in [1] and denoted by C in the text [2] and its earlier editions. It deals with finite binary strings, *strings* for short. Other finite objects can be encoded into single strings in natural ways. The following notions and notation may not be familiar to the reader so we will briefly discuss them. The length of a string x is denoted by $l(x)$. The *empty string* of 0 bits is denoted by ϵ. Thus, $l(\epsilon) = 0$. Let x be a natural number or finite binary string according to the correspondence

$$(\epsilon, 0), (0, 1), (1, 2), (00, 3), (01, 4), (10, 5), (11, 6), \ldots.$$

Then $l(x) = \lfloor \log(x+1) \rfloor$. The Kolmogorov complexity $C(x)$ of x is the length of a shortest string x^* such that x can be computed from x^* by a fixed universal Turing machine (of a special type called "optimal" to exclude undesirable such machines). In this way $C(x)$ is a definite natural number associated with x and a lower bound on the length of a compressed version of it by any known or as yet unknown compression algorithm. We also use the conditional version $C(x|y)$.

The papers by R.J. Solomonoff published in 1964, referenced as [3], contain informal suggestions about the incomputability of Kolmogorov complexity. Says Kolmogorov, "I came to similar conclusions [as Solomonoff], before becoming aware of Solomonoff's work, in 1963–1964." In his 1965 paper [1] Kolmogorov mentioned the incomputability of $C(x)$ without giving a proof: "[. . .] the function $C_\phi(x|y)$ cannot be effectively calculated (generally computable) even if it is known to be finite for all x and y." We give the formal proof of incomputability and discuss recent attempts to compute the Kolmogorov complexity partially, a popular but problematic proposal and some serious options. The problems of the popular proposal are discussed at length while the serious options are primarily restricted to brief citations explaining the methods gleaned from the introductions to the articles involved.

2. Incomputability

To find the shortest program (or rather its length) for a string x we can run all programs to see which one halts with output x and select the shortest. We need to consider only programs of length at most that of x plus a fixed constant. The problem with this process is known as the *halting problem* [4]: some programs do not halt, and it is undecidable which ones they are. A further complication is that we must show there are infinitely many such strings x for which $C(x)$ is incomputable.

The first written proof of the incomputability of Kolmogorov complexity was perhaps in [5] and we reproduce it here following [2] to show what is and what is not proved.

Theorem 1. *The function $C(x)$ is not computable. Moreover, no partial computable function $\phi(x)$ defined on an infinite set of points can coincide with $C(x)$ over the whole of its domain of definition.*

Proof. We prove that there is no partial computable ϕ as in the statement of the theorem. Every infinite computably enumerable set contains an infinite computable subset, see e.g., [2]. Select an infinite computable subset A in the domain of definition of ϕ. The function $\psi(m) = \min\{x : C(x) \geq m, x \in A\}$ is (total) computable (since $C(x) = \phi(x)$ on A), and takes arbitrarily large values, since it can obviously not be bounded for infinitely many x. Also, by definition of ψ, we have $C(\psi(m)) \geq m$. On the other hand, $C(\psi(m)) \leq C_\psi(\psi(m)) + c_\psi$ by definition of C, and obviously $C_\psi(\psi(m)) \leq l(m)$. Hence, $m \leq \log m$ up to a constant independent of m, which is false from some m onward. □

That was the bad news; the good news is that we can approximate $C(x)$.

Theorem 2. *There is a total computable function $\phi(t,x)$, monotonic decreasing in t, such that $\lim_{t\to\infty} \phi(t,x) = C(x)$.*

Proof. We define $\phi(t,x)$ as follows: For each x, we know that the shortest program for x has length at most $l(x) + c$ with c a constant independent of x. Run the reference Turing machine U (an optimal universal one) for t steps on *each* program p of length at most $l(x) + c$. If for any such input p the computation halts with output x, then define the value of $\phi(t,x)$ as the length of the shortest such p, otherwise equal to $l(x) + c$. Clearly, $\phi(t,x)$ is computable, total, and monotonically nonincreasing with t (for all x, $\phi(t',x) \leq \phi(t,x)$ if $t' > t$). The limit exists, since for each x there exists a t such that U halts with output x after computing t steps starting with input p with $l(p) = C(x)$. □

One cannot decide, given x and t, whether $\phi(t,x) = C(x)$. Since $\phi(t,x)$ is nondecreasing and goes to the limit $C(x)$ for $t \to \infty$, if there were a decision procedure to test $\phi(t,x) = C(x)$, given x and t, then we could compute $C(x)$. However, above we showed that C is not computable.

However, this computable approximation has no convergence guarantees as we show now. Let $g_1, g_2, \ldots$ be a sequence of functions. We call f the *limit* of this sequence if $f(x) = \lim_{t\to\infty} g_t(x)$ for all x. The limit is *computably uniform* if for every rational $\epsilon > 0$ there exists a $t(\epsilon)$, where t is a total computable function, such that $|f(x) - g_{t(\epsilon)}(x)| \leq \epsilon$, for all x. Let the sequence of one-argument functions $\psi_1, \psi_2, \ldots$ be defined by $\psi_t(x) = \phi(t,x)$, for each t for all x. Clearly, C is the limit of the sequence of ψs. However, by Theorem 1, the limit is not computably uniform. In fact, by the well-known halting problem, for each $\epsilon > 0$ and $t > 0$ there exist infinitely many x such that $|C(x) - \psi_t(x)| > \epsilon$. This means that for each $\epsilon > 0$, for each t there are many xs such that our estimate $\phi(t,x)$ overestimates $C(x)$ by an *error* of at least ϵ.

3. Computing the Kolmogorov Complexity

The incomputability of $C(x)$ does not mean that we cannot compute $C(x)$ for some xs. For example, if for individual string x we have $C(C(x)|x) = c$ for some constant c, then this means that there is an algorithm of c bits which computes $C(x)$ from x. We can express the incomputability of $C(x)$ in terms of $C(C(x)|x)$, which measures what we may call the "complexity of the complexity function." Let $l(x) = n$. It is easy to prove the *upper bound* $C(C(x)|x)) \leq \log n + O(1)$. However, it is quite difficult to prove the lower bound [6]: For each length n there are strings x of length n such that

$$C(C(x)|x) \geq \log n - \log \log n - O(1)$$

or its improvement by a game-based proof in [7]: For each length n there are strings x of length n such that

$$C(C(x)|x) \geq \log n - O(1).$$

This means that x only marginally helps to compute $C(x)$; most information in $C(x)$ is extra information related to the halting problem.

One way to go about computing the Kolmogorov complexity for a few small values is as follows. For example, let $T_1, T_2, \ldots$ be an acceptable enumeration of Turing machines. Such an acceptable enumeration is a formal concept ([2] Exercise 1.7.6). Suppose we have a fixed reference optimal universal Turing machine U in this enumeration. Let $U(i,p)$ simulate $T_i(p)$ for all indexes i and (binary) programs p.

Run $T_i(p)$ for all i and p in the following manner. As long as i is sufficiently small it is likely that $T_i(p) < \infty$ for all p (the machine T_i halts for every p). The Busy Beaver function $BB(n) : \mathcal{N} \rightarrow \mathcal{N}$ was introduced in [8] and has as value the maximal running time of n-state Turing machines in quadruple format (see [8] or [2] for details). This function is incomputable and rises faster than any computable function of n.

Reference [9] supplies the maximal running time for halting machines for all $i < 5$ and for $i < 5$ it is decidable which machines halt. For $i \geq 5$ but still small there are heuristics [10–13]. A gigantic lower bound for all i is given in [14]. Using Turing machines and programs with outcome the target string x we can determine an upper bound on $C(x)$ for reference machine U (by for each T_i encoding i in self-delimiting format). Please note that there exists no computable lower bound function approximating $C(x)$ since C is incomputable and upper semicomputable. Therefore it cannot be lower semicomputable [2].

For an approximation using small Turing machines we do not have to consider all programs. If I is the set of indexes of the Turing machines and P is the set of halting (or what we consider halting) programs then

$$\begin{aligned}\{(i,p)\}_x = \{(i,p) : T_i(p) = x\} \setminus \{(i,p) : T_i(p) = x \wedge \exists_{i',p'}(T_{i'}(p') = x \\ \wedge |(i,p)| \leq \min\{|i'|, |p'|\}\},\end{aligned}$$

with $i, i' \in I, p, p' \in P$. Here we can use the computably invertible Cantor pairing function [15] which is $f : \mathcal{N} \times \mathcal{N} \rightarrow \mathcal{N}$ defined by $f(a,b) = \frac{1}{2}(a+b)(a+b+1) + b$ so that each pair of natural numbers (a,b) is mapped to a natural number $f(a,b)$ and vice versa. Since the Cantor pairing function is invertible, it must be one-to-one and onto: $|(a,b)| = |a| + |b|$. Here $\{(i,p)\}_x$ is the desired set of applicable halting programs computing x, i.e., if either $|i'|$ or $|p'|$ is greater than some $|(i,p)|$ with $(i,p) \in \{(i,p)\}_x$ while $T_{i'}(p') = x$ then we can discard the pair concerned from $\{(i,p)\}_x$.

4. Problematic Use of the Coding Theorem

Fix an optimal universal prefix Turing machine U. The *Universal distribution* (with respect to U) is $\mathbf{m}(x) = \sum 2^{-l(p)}$ where p is a program (without input) for U that halts. The prefix complexity $K(x)$ is with respect to the same machine U. The complexity $K(x)$ is similar to $C(x)$ but such that the set of strings for which the Turing machine concerned halts is prefix-free (no program is a proper prefix of any other program). This leads to a slightly larger complexity: $K(x) \geq C(x)$. The Coding theorem [16] states $K(x) = -\log \mathbf{m}(x) + O(1)$. Since $-\log \mathbf{m}(x) < K(x)$ (the term $2^{-K(x)}$ contributes to the sum and $2l(x) + O(\log x)$ is also a program for x) we know that the $O(1)$ term is greater than 0.

In [17] it was proposed to compute the Kolmogorov complexity by experimentally approximating the Universal distribution and using the Coding theorem. This idea was used in several articles and applications. One of the last is [18]. It contains errors or inaccuracies for example: "the shortest program" instead of "a shortest program," "universal Turing machine" instead of "optimal universal Turing machine" and so on. Explanation: there can be more than one shortest program, and Turing machines can be universal in many ways. For instance, if $U(p) = x$ for a universal Turing machine, the Turing machine U' such that $U'(qq) = U(q)$ for every q and $U'(r) = 0$ for every string $r \neq qq$ for some string q, is also universal. Yet if U serves to define the Kolmogorov complexity $C(x)$ then U' defines a complexity of x equal to $2C(x)$ which means that the invariance theorem does not hold for Universal Turing machines that are not optimal.

Let us assume that the computer used in the experiments fills the rôle of the required optimal Universal Turing machine for the desired Kolmogorov complexity, the target string, and the universal distribution involved. However, the $O(1)$ term in the Coding theorem is mentioned but otherwise ignored in the experiments and conclusions about the value of the Kolmogorov complexity as reported in [17,18]. Yet the experiments only concern small values of the Kolmogorov complexity, say smaller than 20, so they are likely swamped by the constant hidden in the $O(1)$ term. Let us expand on this issue briefly. In the proof of the Coding theorem, see e.g., [2], a Turing machine T is used to decode a complicated code. The machine T is one of an acceptable enumeration $T_1, T_2, \ldots$ of all Turing machines. The target Kolmogorov complexity K is shown to be smaller than the complexity K_T associated with T plus a constant c representing the number of bits to represent T and other items: $K(x) \leq K_T(x) + c$. Since T is complex since it serves to decode this code, the constant c is huge, i.e., much larger than, say, 100 bits. The values of x for which $K(x)$ is approximated by [17,18] are at most 5 bits, i.e., at most 32. Unless there arises a way to prove the Coding theorem without the large constant c, this method does not seem to work. Other problems: The distribution $\mathbf{m}(x)$ is apparently used as $\mathbf{m}(x) = \sum_{i \in \mathcal{N}, T_i(\epsilon)=x} 2^{-l(\epsilon)}/i$, see ([19] Equation (6)) using a (noncomputable) enumeration of Turing machines $T_1, T_2, \ldots$ that halt on empty input ϵ. Therefore $\sum_{x \in \mathcal{N}} \mathbf{m}(x) = \sum_{i \in \mathcal{N}, T_i(\epsilon) < \infty} 2^{-l(\epsilon)}/i$ and with $l(\epsilon) = 0$ we have $\sum_{x \in \mathcal{N}} \mathbf{m}(x) = \infty$ since $\sum_{x \in \mathcal{N}} 1/x = \infty$. By definition however $\sum_{x \in \mathcal{N}} \mathbf{m}(x) \leq 1$: contradiction. It should be $\mathbf{m}(x) = \sum_{i \in \mathcal{N}, T_i(p)=x} 2^{-l(p)-\alpha(i)}$ with $\sum_{i \in \mathcal{N}} \alpha(i) \leq 1$ as shown in ([2] pp. 270–271).

5. Natural Data

The Kolmogorov complexity of a file is a lower bound on the length of the ultimate compressed version of that file. We can approximate the Kolmogorov complexities involved by a real-world compressor. Since the Kolmogorov complexity is incomputable, in the approximation we never know how close we are to it. However, we assume in [20] that the natural data we are dealing with contain no complicated mathematical constructs like $\pi = 3.1415\ldots$ or Universal Turing machines, see [21]. In fact, we assume that the natural data we are dealing with contains primarily effective regularities that a good compressor finds. Under those assumptions the Kolmogorov complexity of the object is not much smaller than the length of the compressed version of the object.

6. Safe Computations

A formal analysis of the intuitive idea in Section 5 was subsequently and independently given in [22]. From the abstract of [22]: "Kolmogorov complexity is an incomputable function. ... By restricting the source of the data to a specific model class, we can construct a computable function to approximate it in a probabilistic sense: the probability that the error is greater than k decays exponentially with k." This analysis is carried out but its application yielding concrete model classes is not.

7. Short Lists

Quoting from [23]: "Given that the Kolmogorov complexity is not computable, it is natural to ask if given a string x it is possible to construct a short list containing a minimal (plus possibly a small overhead) description of x. Bauwens, Mahklin, Vereshchagin and Zimand [24] and Teutsch [25] show that surprisingly, the answer is YES. Even more, in fact the short list can be computed in polynomial time. More precisely, the first reference showed that one can effectively compute lists of quadratic size guaranteed to contain a description of x whose size is additively $O(1)$ from a minimal one (it is also shown that it is impossible to have such lists shorter than quadratic), and that one can compute in polynomial-time lists guaranteed to contain a description that is additively $O(\log n)$ from minimal. Finally, Ref. [25] improved the latter result by reducing $O(\log n)$ to $O(1)$". See also [26].

8. Conclusions

The review shows that the Kolmogorov complexity of a string is incomputable in general, but may be computable for some arguments. To compute or approximate the Kolmogorov complexity, several approaches have recently been proposed. The most popular of these is inspired by L.A. Levin's Coding theorem and consists of taking the negative logarithm of the so-called universal probability of the string to obtain the Kolmogorov complexity of very short strings (this is not excluded by incomputability as we saw). This probability is approximated by the frequency distributions obtained from small Turing machines. As currently stated, the approach is problematic in the sense that it is only suggestive and cannot be proved correct. Nonetheless, some applications make use of it. Proper approaches either restrict the domain of strings of which the Kolmogorov complexity is desired (so that the incomputability turns into computability) or manage to restrict the Kolmogorov complexity of a string to an item in a small list of options (so that the Kolmogorov complexity has a certain finite probability).

Funding: This research received no external funding.

Acknowledgments: Thank you for CWI's support.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Kolmogorov, A.N. Three approaches to the quantitative definition of information. *Probl. Inform. Transm.* **1965**, *1*, 1–7. [CrossRef]
2. Li, M.; Vitányi, P.M.B. *An Introduction to Kolmogorov Complexity and Its Applications*, 2008 ed.; Springer: Cham, Switzerland; New York, NY, USA, 2008.
3. Solomonoff, R.J. A formal theory of inductive inference, part 1 and part 2. *Inform. Contr.* **1964**, *7*, 224–254. [CrossRef]
4. Turing, A.M. On computable numbers, with an application to the Entscheidungsproblem. *Proc. Lond. Math. Soc.* **1937**, *42*, 230–265; [CrossRef]
5. Zvonkin, A.K.; Levin, L.A. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russ. Math. Surv.* **1970**, *25*, 83–124. [CrossRef]
6. Gács, P. On the symmetry of algorithmic information. *Soviet Math. Dokl.* **1974**, *15*, 1477–1480.
7. Bauwens, B.; Shen, A.K. Complexity of complexity and strings with maximal plain and prefix Kolmogorov complexity. *J. Symbol. Logic* **2014**, *79*, 620–632. [CrossRef]
8. Rado, T. On non-computable functions. *Bell Syst. Tech. J.* **1962**, *44*, 877–884. [CrossRef]
9. Brady, A.H. The determination of the value of Rado's noncomputable function Sigma(k) for four-state Turing machines. *Math. Comp.* **1983**, *40*, 647–665. [CrossRef]
10. Harland, J. Busy beaver machines and the observant otter heuristic (or how to tame dreadful dragons). *Theor. Comput. Sci.* **2016**, *646*, 61–85. [CrossRef]

11. Kellett, O. A Multi-Faceted Attack on the Busy Beaver Problem. Master's Thesis, Rensselaer Polytechnic Institute, Troy, NY, USA, July 2005.
12. Marxen, H.; Buntrock, J. Attacking the Busy Beaver 5. *Bull. EATCS* **1990**, *40*, 247–251.
13. Michel, P. Small Turing machines and generalized busy beaver competition. *Theor. Comput. Sci.* **2004**, *325*, 45–56. [CrossRef]
14. Green, M.W. A lower bound Rado's sigma function for binary Turing machines. In Proceedings of the Fifth Annual Symposium on Switching Circuit Theory and Logical Design, Princeton, NJ, USA, 11–13 November 1964.
15. Cantor's Pairing Function. Available online: https://en.wikipedia.org/wiki/Pairing_function (accessed on 19 February 2020).
16. Levin, L.A. Laws of information conservation (non-growth) and aspects of the foundation of probability theory. *Probl. Inform. Transm.* **1974**, *10*, 206–210.
17. Zenil, H. Une approche expérimentale á la théorie algorithmique de la complexité. Ph.D. Thesis, Lab. d'Informatique Fondamentale de Lille, Université des Sciences et Technologie de Lille, Lille I, Villeneuve-d'Ascq, France, 2011.
18. Soler-Toscano, F.; Zenil, H.; Delahaye, J.P.; Gauvrit, N. Calculating Kolmogorov complexity from the output frequency distributions of small Turing machines. *PLoS ONE* **2014**, *9*, e96223. [CrossRef] [PubMed]
19. Soler-Toscano, F.; Zenil, H. A computable measure of algorithmic probability by finite approximations with an application to integer sequences. *arXiv* **2017**, arXiv:1504.06240.
20. Cilibrasi, R.L.; Vitányi, P.M.B. Clustering by compression. *IEEE Trans. Inf. Theory* **2005**, *51*, 1523–1545. [CrossRef]
21. Vitány, P.M.B. Similarity and denoising. *Philos. Trans. R. Soc. A* **2013**, *371*. doi:10.1098/rsta.2012.0091. [CrossRef] [PubMed]
22. Bloem, P.; Mota, F.; de Rooij, S.; Antunes, L.; Adriaans, P. A safe approximation for Kolmogorov complexity. In Proceedings of the International Conference on Algorithmic Learning Theory, Bled, Slovenia, 8–10 October 2014; pp. 336–350.
23. Zimand, M. Short Lists with Short Programs in Short Time—A Short Proof. In Proceedings of the Conference on Computability in Europe, Budapest, Hungary, 23–27 June 2014.
24. Bauwens, B.; Makhlin, A.; Vereshchagin, N.; Zimand, M. Short lists with short programs in short time. In Proceedings of the 2013 IEEE Conference on Computational Complexity, Stanford, CA, USA, 5–7 June 2013.
25. Teutsch, J. Short lists for shortest programs in short time. *Comput. Complex.* **2014**, *23*, 565–583. [CrossRef]
26. Teutsch, J.; Zimand, M. A brief on short descriptions. *ACM SIGACT News* **2016**, *47*. [CrossRef]