Electronic Colloquium on Computational Complexity, Report No. 125 (2006)

Low-Depth Witnesses are Easy to Find

Luís Antunes, U. Porto
Lance Fortnow, U. Chicago
Alexandre Pinto, U. Porto
André Souto, U.Porto

September 20, 2006

Abstract

Antunes, Fortnow, van Melkebeek and Vinodchandran captured the notion of non-random information by computational depth, the difference between the polynomial-time-bounded Kolmogorov complexity and traditional Kolmogorov complexity We show how to find satisfying assignments for formulas that have at least one assignment of logarithmic depth. The converse holds under a standard hardness assumptions though fails if BPP = UP = EXP.

We also show that under standard hardness assumptions one cannot increase the depth of a string efficiently and that such an assumption is required.

Classification: Computational and structural complexity.

1 Introduction

Kolmogorov (see [LV97]) measures the amount of information in a string by the size of the smallest program that generates that string. If one chooses a string at random this string will have near maximum Kolmogorov complexity even though one can easily create other just as useful random strings using fresh random coins.

To capture non-random information, Bennett [Ben88] defined a notion of logical depth that describes strings with short programs that take a long time to produce their output. Later Antunes, Fortnow, van Melkebeek and Vinodchandran [AFMV06] defined a simpler notion by taking the difference between polynomial-time Kolmogorov complexity and traditional unbounded Kolmogorov complexity. We have seen a number of results about computational depth such as giving a generalization of sparse and random sets [AFMV06] as well as using depth to characterize the worst-case running time of problems that run quickly on average over all polynomial-time samplable distributions [AF05].

In this paper we continue the study of depth. Suppose we have a Boolean formula that has a satisfying assignment of depth d. We show how to probabilistically find an assignment in time exponential in d. We show that under a standard hardness assumption the converse also holds. Under the unlikely but open case that BPP=UP=EXP, one can find formulas that have a single solution of high depth that can be found quickly probabilistically.

We also look at the question of whether one can increase the depth of a string efficiently. We show that under a standard hardness assumption you cannot significantly increase the depth of a string in polynomial time. Once again if BPP=EXP we show examples where one can produce a string of high depth from a string of very low depth.

We also explore the question as to whether a triangle inequality holds for conditional depth.

ISSN 1433-8092

2 Preliminaries

All the strings we use are elements of $\Sigma^* = \{0,1\}^*$. In what follows, the function log denotes $\log_2$ and $|.|$ denotes the length of a string or the cardinality of a set, depending on the context. The letter ε always means the empty string.

2.1 Kolmogorov Complexity

For a full understanding of Kolmogorov complexity we refer the reader to the book of Li and Vitányi [LV97]. We give here only the definitions and results we need. Since our results are not affected by $O(\log n)$ factors that actual model of Kolmogorov complexity is not important. For definition purposes we will use the self-delimiting Kolmogorov complexity. A set of strings A is prefix-free if there are no strings x and y in A where x is a proper prefix of y.

Definition 2.1. *Let U be a fixed Turing machine with a prefix-free domain. For any strings $x, y \in \{0,1\}^*$, the Kolmogorov complexity of x given y is $K(x|y) = \min_p\{|p| : U(p,y) = x\}$.*
For any time constructible t, the t-time-bounded Kolmogorov complexity of x given y is $K^t(x|y) = \min_p\{|p| : U(p,y) = x$ in at most $t(|x|)$ steps$\}$.

The default value for y is the empty string ε, so we typically drop this argument in the notation. We can fix a universal machine U whose program size $|p|$ is at most a constant additive factor worse, and the running time t at most a logarithmic multiplicative factor.
Antunes, Fortnow, van Melkebeek and Vinodchandran [AFMV06] propose a notion of *Computational Depth* as a measure of nonrandom information in a string. Intuitively strings of high depth are low Kolmogorov complexity strings (and hence nonrandom), but a resource bounded machine cannot identify this fact. Indeed, Bennett's logical depth [Ben88] can be viewed as such a measure, but its definition is rather technical. Antunes *et al.* [AFMV06] suggest that the difference between two Kolmogorov complexity measures captures the intuitive notion of nonrandom information. Based on this intuition and with simplicity in mind, in this work we use the following depth measure.

Definition 2.2. *Let t be a constructible time bound. For any string $x \in \{0,1\}^*$,*

$$\text{depth}_t(x) = K^t(x) - K(x)$$

and more generally

$$\text{depth}_t(x|y) = K^t(x|y) - K(x|y).$$

2.2 Pseudo-random generators

Pseudo-random generators are efficiently computable functions which stretch a seed into a long string so that given a random input the output looks random for any resource-bounded machine. Using the hardness of the parity function [Has86], Nisan and Wigderson [NW94] create a pseudo-random generator that looks random to constant depth circuits.

Lemma 2.3 (Nisan-Wigderson)**.** *For any fixed non-negative integer d, there exists a family of generators $\{G_0, G_1, \ldots\}$ with the following properties:*

- *G_n maps strings of length u polynomial in $\log n$ to strings of length n.*

- *For any circuit D of depth d and size n, we have*

$$\left| \Pr_{\rho \in \{0,1\}^n}[D(\rho)] - \Pr_{\sigma \in \{0,1\}^u}[D(G_n(\sigma))] \right| < 1/n.$$

- *Each output bit of G_n is computable in time polynomial in* $\log n$.

The existence of this pseudo-random generator has many applications, in particular, concerning the power of random oracles for classes in PH.
We also use the following result due to Klivans and van Melkebeek [KvM02].

Lemma 2.4 (Klivans-van Melkebeek)**.** *Suppose that for some $\epsilon > 0$, DTIME(2^n) is not contained in DSPACE($2^{\epsilon n}$) for all n then there is a pseudorandom generator mapping $O(\log n)$ bits to n bits that will fool any circuit of n gates even if the circuit has oracle gates for any PSPACE problem.*

When we say assume there is a good pseudorandom generator we mean assume the conclusion of Lemma 2.4.

3 Finding Low-Depth Witnesses

In this section we study the relation between the depth of a solution for a Boolean formula and the existence of a probabilistic algorithm to solve the formula. For this section we assume that n represents the number of variables in the Boolean formula considered.

Proposition 3.1. *Let ϕ be a Boolean formula over n variables. If ϕ has a satisfying assignment w with* $\text{depth}_t(w|\phi) \leq O(\log t + \log n)$ *then there exists a probabilistic algorithm that finds a satisfying assignment of ϕ in time polynomial in t and n.*

Proof. Let $m = K(w|\phi)$. Since $\text{depth}_t(w|\phi) = K^t(w|\phi) - K(w|\phi) \leq c \log t + c \log n$ we can write

$$m' = K^t(w|\phi) \leq m + c \log t + c \log n \tag{3.1}$$

Now consider the following set:

$$A = \{z | \phi(z) = True \text{ and } K^t(z) \leq m'\}$$

where we have used $\phi(z)$ to denote the value of ϕ for the assignment z. The second condition of the definition of A says that if $z \in A$, there exists a program p such that $|p| \leq m'$ and p generates z in time t. By construction, given ϕ and m', A is a computable set and $w \in A$.
From the fact that $w \in A$, we get $m = K(w|\phi) \leq \log |A| + c$. Using the inequality 3.1 we can write:

$$|A| \geq \frac{2^m}{c'} \geq \frac{2^{m'}}{poly(n,t)}.$$

So, the probabilistic algorithm M that produces a satisfiable assignment for ϕ is the following:

Input: Formula ϕ; Output: z an assignment for ϕ;

1. Guess m';
2. Generate randomly a program p of length at most m';

3. Run the universal Turing machine U with program p for t steps;
4. If z is a satisfiable assignment accept, otherwise reject.

Now, the probability of this algorithm generating an assignment for ϕ is at least:

$$\frac{1}{n + c\log t + c\log n} \times \frac{|A|}{2^{m'}} \geq \frac{2^{m'}/poly(n,t)}{2^{m'}} = \frac{1}{poly(n,t)}.$$

□

Consider the converse problem, i.e., if a probabilistic algorithm finds a valid assignment for a Boolean formula ϕ in time t, is there a witness w for ϕ such that $\text{depth}_t(w|\phi) \leq O(\log t + \log n)$, where n is the number of variables occurring in ϕ?
The answer is false if we assume that $BPP = UP = EXP$ and is true if we assume that good pseudo-random generators exist. For the rest of this subsection the time limits will be polynomial in n, i.e., $t = \text{poly}(n)$.

Proposition 3.2. *If $BPP = UP = EXP$, then there exists a Boolean formula ϕ such that for any of its witnesses w produced by a probabilistic algorithm in time t,* $\text{depth}_t(w|\phi) > O(\log t + \log n)$.

Proof. As $BPP = UP = EXP$, then strong one way functions do exist. Let f be a $1-1$ one way function and consider the formula ϕ encoding

$$\exists x : f(x) = y.$$

The string x can be found in exponential time, and by our assumption, this means there exists a probabilistic polynomial time program that produces a valid x for the formula ϕ.
Now, by definition of strong one way function, the shortest program that produces x in polynomial time given ϕ is "**print** x". On the other hand, the shortest program producing x given ϕ has constant length, i.e., $K(x|\phi) = O(1)$, so as t is polynomial in n,

$$\text{depth}_t(x|\phi) = n > O(\log n + \log t).$$

□

We now prove that if good pseudo-random generators exist then the converse proposition holds:

Proposition 3.3. *Let ϕ be a Boolean formula over n variables. If good pseudo-random generators exist and there exists a probabilistic algorithm that produces a witness w of ϕ in time $t(n)$, then* $\text{depth}_t(w|\phi) \leq O(\log n + \log t)$.

Proof. Since there exists a probabilistic algorithm that finds an assignment for ϕ in time $t(n)$, then there exists a random string r such that $|r| < n$ and $\text{depth}_t(w|\phi, r) = O(1)$. Now, given a seed of length $\log r$ we can derandomize the algorithm in BPP by using the pseudo-random generator. So $K^t(w|\phi) \leq O(\log r) \leq O(\log n)$ and then $\text{depth}_t(w|\phi) \leq O(\log n)$. □

4 Depth Cannot Increase Rapidly

In this section we show that if f is an honest polynomial time computable function then it can not significantly increase the depth of its argument, i.e., deep objects are not quickly produced from shallow ones. A function f is honest if for some polynomial p, $p(|f(x)|) \geq |x|$ for all x.
We start showing that it holds for honest efficiently computable functions relative to a random oracle.

Lemma 4.1. *Let $f : \Sigma^* \to \Sigma^*$ be an honest polynomial time computable function and $x \in \Sigma^n$. If $y = f(x)$, then, relative to a random oracle, $\text{depth}_{t'}(y) \leq \text{depth}_t(x) + O(\log n)$, for any polynomial $t(n)$ and some polynomial $t'(n)$ depending on $t(n)$.*

Proof. Consider the following set

$$A_y = \{z : f(z) = y \text{ and } K^t(z) \leq K^t(x)\}$$

By construction, x is in A_y and since f is computable in polynomial time, A_y can be computed by enumerating all programs of size up to $K^t(x)$, running them up to time t and keeping the outputs z that satisfy $f(z) = y$. So,

$$K(x|y) \leq K(A_y|y) + \log |A_y| = \log |A_y| + O(\log n)$$

which implies $|A_y| \geq 2^{K(x|y)+O(\log n)}$. By symmetry of information we have:

$$K(x|y) = K(y|x) + K(x) - K(y) + O(\log n) \tag{4.2}$$

As $y = f(x)$ and f is known, we have that $K(y|x) = O(1)$ and so

$$K(x|y) = K^t(x) - \text{depth}_t(x) - K(y) + O(\log n)$$

implying that

$$|A_y| \geq 2^{K(x|y)+O(\log n)} = \frac{2^{K^t(x)}}{2^{\text{depth}_t(x)+K(y)+O(\log n)}}.$$

So, if we randomly pick a program of size smaller or equal than $K^t(x)$, the probability that it is in A_y is at least

$$\frac{1}{2^{\text{depth}_t(x)+K(y)+O(\log n)}}.$$

Now, considering the set B_y defined as:

$$B_y = \{p : U^t(p) = z, f(z) = y, |p| \leq K^t(x)\}$$

It is easy to see that B_y is computable and any element of B_y is a description of length at most $K^t(x)$ for y. We need to show that for any y there is a program that finds an element of B_y with very high probability.

Any element of A_y has at least one program generating it in B_y so $|B_y| \geq |A_y|$. That means that if we randomly generate a program of length smaller or equal than $K^t(x)$, the probability that it is in B_y is at least

$$|B_y|/2^{K^t(x)} \geq |A_y|/2^{K^t(x)} \geq \frac{1}{2^{\mathrm{depth}_t(x)+K(y)+O(\log n)}}$$

Now let R be a random oracle. We can use R as a function $\Sigma^k \to \Sigma^{K^t(x)}$ where $k = \mathrm{depth}_t(x) + K(y) + O(\log n)$ and then

$$\begin{aligned}
\Pr[\exists w : R(w) \in B_y] &= 1 - \Pr[\forall w : R(w) \notin B_y] \\
&\geq 1 - \left(1 - \tfrac{1}{2^{\mathrm{depth}_t(x)+K(y)+c\log n}}\right)^{2^{\mathrm{depth}_t(x)+K(y)+c'\log n}} \\
&= 1 - e^{-2^{c''\log n}} \\
&= 1 - e^{-n^{c''}} \geq 1 - 2^{-n^{c''}}
\end{aligned}$$

Then

$$\begin{aligned}
\Pr[\forall y \exists w : R(w) \in B_y] &= 1 - \Pr[\exists y \forall w_y : R(w_y) \notin B_y] \\
&\geq 1 - 2^{-n}2^{-n^c} \\
&= 1 - 2^{-n^{c+1}}
\end{aligned}$$

So, for all y, $K^{t'}(y) \leq |w_y| + O(1) = \mathrm{depth}_t(x) + K(y) + O(\log n)$, where t' is the polynomial of the running time of the function determined by the random oracle. From this we conclude that

$$\mathrm{depth}_{t'}(y) \leq \mathrm{depth}_t(x) + O(\log n).$$

□

We now improve the previous result to more general terms by using the composition of the pseudorandom generator in Lemma 2.4 with the pseudorandom generator in Lemma 2.3, as done in [AF05].

Theorem 4.2. *Let $f : \Sigma^* \to \Sigma^*$ be a polynomial time computable function and $x \in \Sigma^n$. If there is a good pseudorandom generator and $y = f(x)$ then $\mathrm{depth}_{t'}(y) \leq \mathrm{depth}_t(x) + O(\log n)$ for any polynomial $t(n)$ and some polynomial $t'(n)$ depending on $t(n)$.*

Proof. Consider the sets A_y and B_y defined as in the proof of Lemma 4.1. The language

$$L = \{\text{for all } y \text{ exists } w_y \text{ such that } f(w_y) \in B_y\}$$

is computable inside the polynomial-time hierarchy with access to a random oracle R. Let M be the oracle Turing machine solving L and denote by R the random oracle. Since M runs in polynomial time, $t(n)$, and has at most $2^{t(n)}$ different paths, it asks at most $O(2^{\mathrm{poly(n)}})$ questions. We can view the computation as a constant-depth circuit over the queries to the oracle and using Lemma 2.3 we can replace the random oracle by polynomially-long random seed. Given a good pseudorandom generator (Lemma 2.4) we can use an $O(\log n)$ bit random string to generate the seed for the first generator. We call the result of the composition of the two pseudorandom genarators G.
Composing this procedure with the procedure of the previous theorem, we have a way to describe y by the following program for a fixed y:

Input: a seed s and a witness w_y

Output: y

1. Compute $s' = G(s)$.
2. Compute $h = R(s')$ where R is a procedure that computes a function h from the oracle s' as per Lemma 4.1. The result is a function that maps a seed of length $\text{depth}_t(x) + K(y) + O(\log n)$ into a program of size at most m.
3. Compute $p = h(w_y)$, giving a program in the set B_y.
4. Run the universal Turing Machine with program p and call the output z.
5. Compute $f(z)$. (By the construction in the Lemma 4.1, this is y.)
6. Output y.

Since the several constructions are independent of any given instance, we have a description for y requiring only the description of s and w_y that can be computed in time $t'(n)$, a polynomial depending on running time of the function h, depending on t, and the running time of p, again depending on t. Therefore,

$$\begin{aligned} K^{t'}(y) &\leq |s| + |w_y| + O(1) \\ &= O(\log n) + d + K(y) \end{aligned}$$

So, $\text{depth}_{t'}(y) \leq \text{depth}_t(x) + O(\log n)$.

□

However if BPP = EXP the previous result do not hold.

Theorem 4.3. *Assume* BPP = EXP. *There exists a polynomial-time computable function f such that for all polynomials t and t' there exists a polynomial q such that for every natural number n there exists strings y and x with $|y| = n$ and $|x| = q(n)$ such that*

1. $y = f(x)$,
2. $\text{depth}_t(y) \geq n - O(\log n)$, *and*
3. $\text{depth}_{t'}(x) \leq O(\log n)$.

Proof. Fix n. Let y be the lexicographically least string such that $K^t(y) \geq n$. We can compute y so $K(y) \leq O(\log n)$ and $\text{depth}_t(y) \geq n - O(\log n)$.
We can find y in time $2^{O(n)}$ given n and $t(n)$ so by the hypothesis BPP = EXP there is a probabilistic algorithm A that will efficiently compute y. Note that the running time of A is independent of t'.
Let $m = q(n)$ be the number of random bits used by A. Let $f(r)$ simulate A using r as the random coins. Let x be Kolmogorov random of length m.
We have $f(x) = y$ since the set of strings that cause f to give the wrong answer will be small and all such strings will have low Kolmogorov complexity.
Finally we have $\text{depth}_{t'}(x) \leq O(\log n)$ because x is random.

□

5 Properties of Conditional Depth

Bennett [Ben88] noted that the impossibility of rapid growth of depth can not be extended to a transitive law relative to shallowness, i.e., if x is shallow relative to y and y is shallow relative to z, this does not necessarily imply that x is shallow relative to z. Bennett considered z do be a Kolmogorov random string of size n, $y = 0^n$ and $x = z \oplus d$ where d is some deep string. Bennett's example does not work for computational depth. It is easy to see that $K(z \oplus d|z)$ is small however when looking at $K^t(z \oplus d|z) = K^t(d|z)$, if $BPP = EXP$ we may use z to find d.
We conjecture that the transitive law relative to shallowness does not hold. However, assuming that pseudo-random generators exist, we show that depth satisfies an analog of a triangular inequality.

Theorem 5.1. *Let $x, y, z \in \{0,1\}^*$ and $n = \max(|x|, |y|, |z|)$. Given a polynomial $t(n)$ and assuming the existence of pseudo-random generators, then there exists a polynomial $t'(n)$ such that*

$$\text{depth}_{t'}(y|z) \leq \text{depth}_t(x|z) + \text{depth}_t(y|x,z) + O(\log n)$$

Proof. Define $a = K^t(x|z)$, $b = K^t(y|x,z)$, $\text{depth}_t(x|z) = r$ and $\text{depth}_t(y|x,z) = s$. Then,

$$K(x|z) = a - r \text{ and } K(y|x,z) = b - s.$$

Consider the following set:

$$A = \{w : \text{ exists } u \text{ exists } v \text{ } s.t. \text{ } |u| \leq a,\ U^t(u,z) = w,\ |v| \leq b \text{ and } U^t(v,w) = y\}.$$

By construction, x is an element of A and A is computable given y and z. Then, $K(x|y,z) \leq \log|A| + c$, i.e., A has at least $2^{K(x|y,z)-c}$ elements. By symmetry of information, we have that

$$\begin{aligned} K(x|y,z) &= K(y|x,z) + K(x|z) - K(y|z) + O(\log n) \\ &= b - s + a - r - K(y|z) + O(\log n) \end{aligned}$$

Thus

$$|A| \geq \frac{2^{a+b+O(\log n)}}{2^{r+s+K(y|z)+O(\log n)}}.$$

Then, the probability of a random program p, of size smaller than $a+b+O(\log n)$, generating y is at least $\frac{1}{2^{r+s+K(y|z)+O(\log n)}}$. Using a similar construction of Theorem 4.2 we can find a seed of size $r + s + K(y|z) + O(\log n)$ that generates a program producing y within polynomial time t'. So,

$$K^{t'}(y|z) \leq K(y|z) + r + s + O(\log n)$$

and then

$$\text{depth}_{t'}(y|z) \leq r + s + O(\log n).$$

□

There is an easy consequence of this theorem showing that the computational depth is independent of its environment. Formally we have:

Corollary 5.2. *Let $x, y \in \{0,1\}^*$, $n = \max(|x|, |y|)$, $\text{depth}_t(x) = r$ and $\text{depth}_t(y|x) = s$ for some polynomial $t(n)$. Then, assuming the existence of pseudo-random generators there exists a polynomial $t'(n)$ such that*

$$\text{depth}_{t'}(y) \leq r + s + O(\log n).$$

However if we replace the pseudo-random generators assumption by the assumption BPP = EXP the previous results do not hold.

Theorem 5.3. *If* BPP = EXP*, then there exist* $x, y \in \{0,1\}^*$ *such that* $\text{depth}_t(x)$ *and* $\text{depth}_t(y|x)$ *are small but* $\text{depth}_t(y)$ *is big, with* $n = \max(|x|, |y|)$ *and* t *a polynomial in* n*.*

Proof. Let x be a Kolmogorov random string, and y the least lexicographic string with high $K^t(y)$. Since all random strings are shallow, $\text{depth}_t(x)$ is small.
The string y can be computed by the following procedure: enumerate all binary strings in lexicographic order and for each string w, compute $K^t(w)$. If this number is bigger than a certain threshold, then output w and stop. Then, $K(y) = O(1)$. On the other hand, by construction $K^t(y)$ is high, so $\text{depth}_t(y)$ is large.
Now we address $\text{depth}_t(y|x)$. Since $K(y)$ is small, then $K(y|x)$ is also small. Define L as the language consisting exactly of the string y. The program for y given above enumerates at most $2^{|y|}$ strings before outputting a result, and for each of them it executes up to a polynomial number of steps no more than $2^{K(y)} \leq 2^{|y|}$ programs. Thus, $L \in$ EXP. Since by assumption BPP = EXP, there is a probabilistic algorithm that takes a certain random input, runs in polynomial time and outputs y. We let this random input be x. Taking the size of the probabilistic algorithm to be a constant, $K^t(y) = O(1)$ and thus $\text{depth}_t(y|x) = O(1)$.

□

Acknowledgments

We thank Harry Buhrman and Adam Harrow for helpful discussions.

References

[AFV01] L. Antunes and L. Fortnow. Sophistication Revisited. *Proceedings of the 30th International Colloquium on Automata, Languages and Programming*, volume 2719 of Lecture Notes in Computer Science, pages 267-277. Springer, 2003.

[AF05] L. Antunes and L. Fortnow. Time-bounded universal distributions. Technical Report TR05-144, Electronic Colloquium on Computational Complexity, 2005.

[AFMV06] L. Antunes and L. Fortnow and D. van Melkebeek and N. V. Vinodchandran. Computational depth: concept and applications. *Theor. Comput. Sci.*, 354 (3): 391–404, 2006.

[Ben88] Bennett,C. Logical Depth and Physical Complexity *The Universal Turing Machine, A Half-Century Survey*, pages: 227-257. Oxford University Press, 1988.

[Cha66] G. J. Chaitin. On the length of programs for computing finite binary sequences. *Journal of the ACM*, 13(4):145–149, 1966.

[Has86] J. Hastad. Computational limitations of small-depth circuits. *Ph. D. Thesis*, MIT Press, 1986.

[KvM02] A. Klivans and D. van Melkebeek. Graph Nonisomorphism Has Subexponential Size Proofs Unless The Polynomial-Time Hierarchy Collapses. SIAM Journal on Computing, 31: 1501-1526, 2002.

[Kol65] A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems Inform. Transmission*, 1(1):1–7, 1965.

[LV97] Ming Li and Paul M. B. Vitányi. *An introduction to Kolmogorov complexity and its applications*. Springer Verlag, 2nd edition, 1997.

[NW94] N. Nisan and A. Wigderson. Hardness vs. randomness *J. Comput. Syst. Sci*, 49 (2):149–167, 1994

[Sip83] M. Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th ACM Symposium on the Theory of Computing*, pages 330–335, 1983.

[Sol64] R. Solomonoff. A formal theory of inductive inference, part I. *Information and Control*, 7(1):1–22, 1964.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/