LOW-DEPTH WITNESSES ARE EASY TO FIND

Luís Antunes, Lance Fortnow, Alexandre Pinto, and André Souto

Abstract. Kolmogorov Complexity measures the amount of information in a string by the size of the smallest program that generates that string. Antunes, Fortnow, van Melkebeek and Vinodchandran captured the notion of useful information by computational depth, the difference between the polynomial-time-bounded Kolmogorov complexity and traditional Kolmogorov complexity.
We show unconditionally how to probabilistically find satisfying assignments for formulas that have at least one assignment of logarithmic depth. The converse holds under a standard hardness assumption though fails if **BPP** = **FewP** = **EXP**. We also prove that assuming the existence of good pseudorandom generators one cannot increase the depth of a string efficiently.

Keywords. Computational depth, SAT formulas, Kolmogorov complexity, pseudorandom generators

Subject classification. 68Q17, 68Q30

1. Introduction

Kolmogorov (1965), Solomonoff (1964) and Chaitin (1966) independently defined the complexity of a string x as the length of the shortest program that produces x. 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 string using fresh random coins.

To capture useful information, Bennett (1988) formally defined the *s-significant logical depth* of an object x as the time required by a standard universal Turing machine to generate x by a program that is no more than s bits longer than the shortest descriptions of x. Later, Antunes *et al.* (2006) defined a simpler notion by taking the difference between polynomial-time Kolmogorov complexity

and traditional unbounded Kolmogorov complexity. Intuitively, computational depth measures the amount of non-random or useful information in a string. We have seen a number of results about computational depth such as giving a generalization of sparse and random sets (Antunes *et al.* (2006)) 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 (Antunes & Fortnow (2009)).

In this paper we continue the study of computational depth. Suppose we have a Boolean formula that has a satisfying assignment of computational 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 $\mathbf{BPP} = \mathbf{FewP} = \mathbf{EXP}$, one can find formulas that have few solutions, all of them of high computational depth, that can be found in probabilistic polynomial time.

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, exponential time is not infinitely often in subexponencial space, one cannot significantly increase the depth of a string in polynomial time. Once again, if $\mathbf{BPP} = \mathbf{EXP}$, we show examples where one can produce a string of high depth from a string of very low depth. Finally, we explore the question as to whether a triangle inequality holds for conditional computational depth.

The rest of this paper is organized as follows. In the next section, we present notations and definitions used. We also state some related results from the literature. In Section 3, we study the relation between computational depth of a solution for a Boolean formula and the existence of a probabilistic algorithm to solve that formula. In Section 4, we show that the computational depth cannot increase rapidly and finally in Section 5, we study some properties of computational depth.

2. Preliminaries

All the strings used are elements of $\Sigma^* = \{0,1\}^*$. 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.

The complexity class **BPP** consists of problems computable in probabilistic polynomial time with two-sided error bounded away from one-half. The class **EXP** is the set of problems computable in time $2^{p(n)}$ for some polynomial $p(n)$. The class **FewP** is the set of problems accepted by a polynomial-time nondeterministic Turing machine that has at most a fixed polynomial number

of accepting paths for each input.

2.1. Kolmogorov Complexity. For a full understanding of Kolmogorov complexity we refer the reader to the book of Li & Vitányi (2008). Here, we present just the definitions and results needed. 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. *A function $t : \mathbf{N} \to [0, \infty)$ is* time-constructible *if there exists a Turing machine that runs in time exactly $t(n)$ on every input of size n.*

All explicit resource bounds we use in this paper are time-constructible.

DEFINITION 2.2. *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 and we typically drop this argument in the notation. We fix a universal prefix-free machine U. Notice that a different reference universal prefix-free machine may affect the program sizes $|p|$ by at most a constant additive term, and the running times t by at most a logarithmic multiplicative term. So, in this sense, Kolmogorov complexity theory is machine independent.

DEFINITION 2.3. *A string x is* incompressible *if $K(x) \geq |x|$. We also call such x algorithmically random.*

2.2. Computational Depth. The Kolmogorov complexity of a string x does not take into account the time necessary to produce the string from a description of length $K(x)$. Time-bounded Kolmogorov complexity gives us exactly this. However it may be possible that the smallest program is just a few bits smaller than a much faster program. Levin (1973) introduced a useful variant of Kolmogorov complexity weighing program size with the penalty of its running time.

DEFINITION 2.4. *For any strings x, y, the Levin complexity of x given y is*

$$Ct(x|y) = \min_p\{|p| + \log t : U(p, y) \text{ halts in at most } t \text{ steps and outputs } x\}.$$

After some attempts, Bennett (1988) formally defined the s-significant logical depth of a string x as the time required by a standard universal Turing machine to generate x by a program that is no more than s bits longer than the shortest descriptions of x. A string x is called logically deep if it takes a relatively large amount of time to generate it from any short description.

DEFINITION 2.5. *Let x be a string and s be a nonnegative integer. The logical depth of x at a significance level s is*

$$\text{depth}_s(x) \quad = \quad \min\left\{t(|x|) : \frac{\sum_{p:U^t(p)=x} 2^{-|p|}}{2^{-K(x)}} \geq 2^{-s}\right\}$$

Note that algorithmically random strings are shallow at any significance level. In particular, Chaitin's Ω is shallow. Deep strings are hard to find, however they can be constructed by diagonalization, see Bennett (1988).

Antunes *et al.* (2006) proposed a notion of *Computational Depth* as a measure of nonrandom information in a string. Intuitively, strings of high computational depth are low Kolmogorov complexity strings (and hence nonrandom), but a resource bounded machine cannot identify this fact. Indeed, Bennett's logical depth (Bennett (1988)) can be viewed as such a measure, but its definition is rather technical. Antunes *et al.* (2006) 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 2.6. *Let t be a constructible time bound. For any strings $x, y \in \{0,1\}^*$,*

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

and more generally,

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

2.3. Pseudorandom generators. Pseudorandom generators are efficiently computable functions which stretch a seed into a long string so that for a random input the output looks random for a resource-bounded machine. We need some pseudorandom generators based on hard functions.

The following lemma follows from Nisan & Wigderson (1994).

LEMMA 2.7 (Nisan-Wigderson). *Suppose we have a set $\mathcal{H}$ of functions from Σ^k to Σ^m, a polynomial p and a parameter n with $\log(m) \leq k \leq p(n)$ with the following properties.*

(i) At least 3/4 of all possible functions mapping Σ^k to Σ^m are in $\mathcal{H}$, and

(ii) For some a, there is a $\Sigma_a^{p(n)}$-machine with oracle access to a function H which on input 1^n will accept exactly when H is in $\mathcal{H}$.

Then there is a function $H'(x,r)$ with $x \in \Sigma^k$ and $|r|$ polynomial in n, such that each output bit is computed in polynomial time (in n) and for at least 2/3 of the possible r, $\hat{H}_r(x) = H'(x,r)$ is in $\mathcal{H}$.

PROOF. View a function H as a binary string of length $M = m2^k$ and the $\Sigma_a^{p(n)}$-machine as a constant depth circuit C of size $2^{n^{O(1)}}$. Nisan & Wigderson (1994) show how to create a pseudorandom generator G, based on the parity function, that maps a seed of size polynomial in n to M bits that fools C. Each output bit of the generator G can be computed in time polynominal in n. H' is easily constructed from G. □

Impagliazzo & Wigderson (1996) strengthen the work of Nisan and Wigderson to show how to achieve full derandomization based on strong hardness assumptions. Klivans & van Melkebeek (2002) generalize Impagliazzo-Wigderson by showing the results hold for relativized worlds in a strong way.

LEMMA 2.8 (Impagliazzo-Wigderson, Klivans-van Melkebeek). *For any oracle A, suppose that there are languages in $\mathbf{DTIME}(2^{O(n)})$ that for some $\epsilon > 0$, cannot be computed by circuits of size $2^{\epsilon n}$ with access to an oracle for A. Then there is a k and a pseudorandom generator $g : \Sigma^{k \log n} \to \Sigma^n$ computable in time polynomial in n such that for all relativizable circuits C of size n*

$$\left| \Pr_{s \in \Sigma^{k \log n}}(C^A(g(s)) = 1) - \Pr_{r \in \Sigma^n}(C^A(r) = 1) \right| = o(1).$$

We need the following hardness hypothesis for many of the derandomization results in our paper.

HYPOTHESIS 2.9. *There is a language L in $\mathbf{DTIME}(2^{O(n)})$ that for some $\epsilon > 0$, L cannot be computed on infinitely many input lengths by circuits of size $2^{\epsilon n}$ with Σ_2^p gates.*

Miltersen (2001) shows that Hypothesis 2.9 follows from a uniform statement of time versus space. He showed that if $\mathbf{DTIME}(2^{O(n)})$ is not contained in $\mathbf{DSPACE}(2^{o(n)})$ then $\mathbf{DTIME}(2^{O(n)})$ contains a language that does not have circuits of size $2^{o(n)}$ with Σ_2^p gates or even **PSPACE** gates.

LEMMA 2.10 (Miltersen). *If* $\mathbf{DTIME}(2^{O(n)})$ *is not contained in* $\mathbf{DSPACE}(2^{o(n)})$ *for infinitely many input lengths then for every language* A *in* **PSPACE** *there is some* $\epsilon > 0$ *such that* $\mathbf{DTIME}(2^{O(n)})$ *contains a language that does not have circuits of size* $2^{\epsilon n}$ *with access to* A *for infinitely many input lengths.*

3. Finding Low-Depth Witnesses

In this section we study the relation between computational 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.

THEOREM 3.1. *Let* c *be a constant and* t *some polynomial. There is a probabilistic polynomial time algorithm such that, if* ϕ *is a Boolean formula over* n *variables with a satisfying assignment* w *of* $\mathrm{depth}_t(w|\phi) \leq c \log n$*, then on input* ϕ *the algorithm outputs a satisfying assignment of* ϕ *with probability close to one.*

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

$$m' = K^t(w|\phi) \leq m + c \log n \tag{3.2}$$

Now consider the following set:

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

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 $m = K(w|\phi) \leq K(w|\phi, m') + K(m') \leq \log |A| + O(\log n)$. Using Inequality (3.2) we can write, for some constant c' depending on t and on c:

$$|A| \geq \frac{2^m}{c'n} \geq \frac{2^{m'}}{poly(n)}.$$

where $poly(n) = c'n^{c'}$. The following probabilistic algorithm M produces a satisfiable assignment for ϕ.

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

1. Guess $m' \leq n$;
2. Generate randomly a program p of length at most m';
3. Run the universal Turing machine U with program p and the formula ϕ for t steps and let z be the output;
4. If z is a satisfiable assignment accept, otherwise reject.

The probability of this algorithm generating an assignment for ϕ is at least:

$$\frac{1}{n + c\log n} \times \frac{|A|}{2^{m'}} \geq \frac{2^{m'}/poly(n)}{2^{m'}} = \frac{1}{poly(n)}.$$

If we run the algorithm a polynomial number of times, with high probability, one of those runs will produce a satisfying assignment of ϕ. □

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 $\text{depth}_t(w|\phi) \leq O(\log n)$, where n is the number of variables occurring in ϕ?

The answer is false if we assume that **BPP** = **FewP** = **EXP** and is true if we assume that good pseudorandom generators exist.

THEOREM 3.3. *If* **BPP** = **FewP** = **EXP** *then we can find a witness for every satisfiable formula in probabilistic polynomial time but for every polynomial q there exists infinitely many Boolean formulas ϕ and some $\epsilon > 0$ such that*

$$\text{depth}_q(w|\phi) \geq n^\epsilon$$

for all satisfying assignments w of ϕ.

PROOF. For any formula ϕ we can find a satisfying assignment in exponential time and since **EXP** = **BPP** we can find a witness probabilistically quickly.

Fix a complete language L for **EXP** and since L is in **FewP**, let M be an **NP** machine accepting L with at most $p(n)$ accepting paths on each input.

Let x be in L with $|x| = n$. By Cook's reduction of the **NP**-completeness of SAT, we can compute a ϕ that has the same number of satisfying assignments as $M(x)$ has accepting computations. Let w be any witness of ϕ. We have

- $K(w|\phi) = O(\log n)$, since we can search for the satisfying assignments w of ϕ and identify one of them by its index.
- Let q be any polynomial. If $K^q(w|\phi) \leq n^{o(1)}$ then we can compute whether x is in L in deterministic time $2^{n^{o(1)}}$ by trying all small programs and seeing if any of them produce a witness.

By the time hierarchy theorem **EXP** is not contained in deterministic time $2^{n^{o(1)}}$ so there must be infinitely many ϕ with $K^q(w|\phi) \geq n^\delta$ for some $\delta > 0$. Thus we get $\text{depth}_q(w|\phi) \geq n^\epsilon$ for any $\epsilon < \delta$. □

We now prove that if good pseudorandom generators exist then the converse proposition holds, i.e., all the satisfying assignments w of ϕ with n variables produced by a probabilistic algorithm in time $t(n)$ have $\text{depth}_t(w|\phi) \leq O(\log n + \log(t(n)))$.

PROPOSITION 3.4. *Let c be a constant, t some polynomial and A a probabilistic algorithm that on an input Boolean formula with n variables runs in time $t(n)$. Under Hypothesis 2.9, for any n and formula ϕ over n variables if the algorithm outputs a satisfying assignment of ϕ with probability at least $2/3$ then there is a satisfying assignment w of ϕ with* $\text{depth}_t(w|\phi) \leq c(\log n + \log t)$.

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

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 computational depth of its argument, i.e., deep objects are not quickly produced from shallow ones. A function f is honest if for some k, $|f(x)| \geq |x|^{1/k}$ for all x.

We start by showing that relative to a random oracle for every honest efficiently computable f, the depth of $f(x)$ cannot be much greater than the depth of x. Later on we will replace the random oracle with a pseudorandom generator.

We say that a statement holds relative to a random oracle if we choose an oracle R uniformly at random, and with probability one the statement is true when all computations have access to the oracle R.

LEMMA 4.1. *For most oracles R the following holds: If $f : \Sigma^* \to \Sigma^*$ is an honest polynomial-time computable function relative to R then for any polynomial $t(n)$ there is a polynomial $t'(n)$ so that* $\mathrm{depth}_{t'}(f(x)|f,R) \leq \mathrm{depth}_t(x|f,R) + O(\log n)$ *for all $x \in \Sigma^*$.*

PROOF. Fix x, let $y = f(x)$ and define the set A_y as the set of strings $z' \in \Sigma^{K^t(x|f,R)}$ for which there is a prefix z of z' such that $U^{f,R}(z) = x'$ in time t and $f(x') = y$.

By construction, x can be computed from a string in A_y (in fact, from x^* the first string in lexicographic order that, with oracle access to f and R, produces x in time t). Since f is computable in polynomial time in R, A_y can be computed (given $K^t(x|f,R)$ and y) by enumerating all programs of size up to $K^t(x|f,R)$, if a program z outputs a string x' in time t with $f(x') = y$ then we output all $z' \in \Sigma^{K^t(x|f,R)}$ that extend z. We then have

$$K(x|y,R) \leq K(\chi_{A_y}|y,f,R) + \log|A_y| + O(\log n) = \log|A_y| + O(\log n)$$

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

$$K(x|y,f,R) = K(y|x,f,R) + K(x|f,R) - K(y|f,R) \pm O(\log n).$$

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

$$K(x|y,f,R) = K^t(x|f,R) - \mathrm{depth}_t(x|f,R) - K(y|f,R) \pm O(\log n)$$

implying that

$$|A_y| \geq 2^{K(x|y,f,R) - O(\log n)} = \frac{2^{K^t(x|f,R)}}{2^{\mathrm{depth}_t(x|f,R) + K(y|f,R) + c\log n}}$$

for some constant c.

If we randomly pick a string of size $K^t(x|f,R)$, the probability that it is in A_y is at least

$$\frac{1}{2^{\mathrm{depth}_t(x|f,R) + K(y|f,R) + c\log n}}.$$

Since f is honest there is some constant q such that $|z| \leq |y|^q$ for all z such that $f(z) = y = f(x)$. Let $m = |y|^q + 1$. Note that since f is polynomial-time computable and honest, m is bounded by a polynomial in $|x|$.

Let c' be a constant to be specified later. Define a function $g : \Sigma^k \to \Sigma^\ell$ where $k = \mathrm{depth}_t(x|f,R) + K(y|f,R) + c'\log n$ and $\ell = K^t(x|f,R)$, by letting

the jth bit of the output of $g(w)$ be $R(\langle 1^m, w, j\rangle)$. When R is chosen at random the output of $g(y)$ is completely independent from the part of R used to define A_y since A_y depends on strings in R only of length less than m.

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

Then

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

For the appropriate choice of c', with high probability, for every y we can find a w_y such that $g(w_y)$ is in A_y, i.e., we can find a w_y for which there is a prefix z of $g(w_y)$ satisfying $f(U(z)) = y$. So, for some polynomial t' we have

$$K^{t'}(y|f,R) \leq |w_y| + O(\log n) = \mathrm{depth}_t(x|f,R) + K(y|f,R) + O(\log n).$$

and thus conclude

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

By the Kolmogorov zero-one law (see Kolmogorov (1950)) this high probability result implies these statements must hold over the choice of R with probability one. □

Under a standard hardness assumption, exponential time is not infinitely often in subexponential space, we now improve the previous result to more general terms. The idea behind the proof is to compose the pseudorandom generator in Lemma 2.8 with the pseudorandom generator in Lemma 2.7, as done by Antunes & Fortnow (2009).

THEOREM 4.2. *Let $f : \Sigma^* \to \Sigma^*$ be an honest polynomial time computable function and $x \in \Sigma^n$. Then, under Hypothesis 2.9, for any polynomial $t(n)$ there is a polynomial $t'(n)$ so that* $\mathrm{depth}_{t'}(y|f) \leq \mathrm{depth}_t(x|f) + O(\log n)$, *where* $y = f(x)$.

PROOF. Since depth_t decreases as t increases, we assume without loss of generality that t is much larger than the running time of f.

Notice that Theorem 4.2 is true without any assumptions if $K(f(x)) \geq K(x) - O(\log n)$ as, $\text{depth}_t(f(x)|f) = K^t(f(x)|f) - K(f(x)|f) \leq K^t(f(x)|f) - K(x|f) + O(\log n) \leq K^t(x|f) - K(x|f) + O(\log n) = \text{depth}_t(x|f) + O(\log n)$. So we will focus on the case $K(f(x)) < K(x) - c \log n$ for all constant c.

Continuing the proof of Lemma 4.1, consider the set $\mathcal{H}$ of functions $h : \Sigma^k \to \Sigma^m$ where $k = \text{depth}_t(x|f) + K(y|f) + c' \log n$ and $m = K^t(x|f)$ such that for all y there is a w_y such that for some prefix z of $h(w_y)$, $f(U(z)) = y$. Notice

1. By the same argument as in the proof of Lemma 4.1, a randomly chosen h will fall into $\mathcal{H}$ with probability very close to one.
2. By definition of $\mathcal{H}$, we can determine whether h sits in $\mathcal{H}$ in $\Pi_2^{p(n)} \subseteq \Sigma_3^{p(n)}$ with oracle access to h.

With the above conditions and the fact that $\log m < k < 3n$ for sufficiently large c', the set $\mathcal{H}$ fulfills the requirements of Lemma 2.7 so we can use a polynomially-long random seed to describe an h in $\mathcal{H}$. Given a good pseudo-random generator (Lemma 2.8) 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 generators 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. Now consider s' as a function h that maps a sequence of length $\text{depth}_t(x) + K(y) + O(\log n)$ into a program of size at most $K^t(x)$, as per Lemma 4.1.
3. Compute $p = h(w_y)$, giving a program in the set A_y.
4. Run the universal Turing Machine with the appropriated prefix of p and call the output x'.
5. Compute $f(x')$. (By the construction in the Lemma 4.1, this is y.)
6. Output y.

Since all constructions are independent of any given instance, we have a description for y requiring only the description of s, w_y and an $O(\log n)$ term to account for the information needed to take the prefix, that can be computed in time $t'(n)$, a polynomial depending on the running time of the function h, depending on t, and the running time of p, again depending on t. Therefore,

$$\begin{aligned} K^{t'}(y|f) &\leq |s| + |w_y| + O(\log n) \\ &= O(\log n) + \mathrm{depth}_t(x|f) + K(y|f) \end{aligned}$$

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

However if **BPP** = **EXP** the previous result does not hold.

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

(i) $y = f(x)$,

(ii) $\mathrm{depth}_t(y) \geq n - O(\log n)$, *and*

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

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 $\mathrm{depth}_t(y) \geq n - O(\log n)$.

We can find y in time $2^{O(n)}$ given n and $t(n)$ so by the hypothesis **BPP** = **EXP** there is a probabilistic polynomial time algorithm A that will compute y given 1^n.

Let $m = q(n) > n$ be the number of random bits used by A. Let $f(r)$ simulate A using r as the random coins on input 1^n where n is derived from the length of r. 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 $\mathrm{depth}_{t'}(x) \leq O(\log n)$ because x is random. □

5. Properties of Conditional Depth

Bennett (1988) 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. We conjecture this transitive law does not hold for computational depth either but this question remains open.

However, assuming that pseudorandom generators exist, we show that depth satisfies an analog of a triangular inequality, informally that the depth of y is bounded by the depth of x and the depth of y given x. We actually prove something slightly stronger (Corollary 5.2).

THEOREM 5.1. *Under the Hypothesis 2.9, given a polynomial $t(n)$ there exists a polynomial $t'(n)$ such that for any $x, y, z \in \{0,1\}^*$ and $n = \max(|x|, |y|, |z|)$*

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

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

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

Consider the set A consisting of all w such that there exists a $|u| \leq a$ and $|v| \leq b$ with $U^t(u,z) = w$ and $U^t(v,w,z) = y$, i.e.,

$$A = \left\{ w | K^t(w|z) \leq a \wedge K^t(y|w,z) \leq b \right\}.$$

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

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

Thus

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

for some constant c'. The probability of a random program p of size smaller than $a+b+O(\log n)$ generating y is at least $\frac{1}{2^{r+s+K(y|z)+c' \log n}}$. 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

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

□

Under the Hypothesis 2.9, given a polynomial $t(n)$ there exists a polynomial $t'(n)$ such that for any $x, y, z \in \{0,1\}^*$ and $n = \max(|x|, |y|, |z|)$

COROLLARY 5.2. *Under the Hypothesis 2.9, given a polynomial $t(n)$ there exists a polynomial $t'(n)$ such that for any $x, y \in \{0,1\}^*$ and $n = \max(|x|, |y|)$*

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

However if we replace the pseudorandom generators assumption by the assumption **BPP** = **EXP** the previous results do not hold. In fact, in the next result we show a pair of strings x and y that under the assumption **BPP** = **EXP**, satisfy $\mathrm{depth}_t(x)$ and $\mathrm{depth}_t(y|x)$ are small and $\mathrm{depth}_t(y)$ is big.

THEOREM 5.3. *If* **BPP** = **EXP**, *then there exist $x, y \in \{0,1\}^*$ such that $\mathrm{depth}_t(x) \leq O(1)$ and $\mathrm{depth}_t(y|x) \leq O(1)$ but $\mathrm{depth}_t(y) \geq n - O(1)$, with $n = \max(|x|, |y|)$ and t a polynomial in n.*

PROOF. Let x be a Kolmogorov random string, i.e., such that $K(x) \geq |x|$, and y the lexicographically least string satisfying $K^t(y) \geq |y|$. Since all random strings are shallow, $\mathrm{depth}_t(x) \leq O(1)$.

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 $|y|$, then output w and stop. Thus, $K(y) = O(1)$. On the other hand, by construction $K^t(y) \geq |y|$, so $\mathrm{depth}_t(y) \geq |y| - O(1)$.

Now we address $\mathrm{depth}_t(y|x)$. Since $K(y) \leq O(1)$, then $K(y|x) \leq O(1)$. 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. 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 $\mathrm{depth}_t(y|x) = O(1)$. □

Acknowledgements

We thank Harry Buhrman and Aram Harrow for helpful discussions. We thank the anonymous Complexity reviewers for many useful comments, particularly for pointing out that in Theorem 3.3 we could use **FewP** instead of **UP**.
The authors from University of Porto and Minho were partially supported by funds granted to LIACC through the Programa de Financiamento Plurianual, FCT and Programa POSI and are also supported by the project CSI^2 (PTDC/EIA- CCO/099951/2008) granted by FCT to Instituto de Telecomunicações. André Souto was also supported by the grant SFRH/BD/28419/2006 of FCT. Lance Fortnow was supported in part by NSF grants CCF-0829754 and DMS-0652521.

References

L. Antunes & L. Fortnow (2009). Worst-Case Running Times for Average-Case Algorithms. *Proceedings of Annual IEEE Conference on Computational Complexity* 298–303.

L. Antunes, L. Fortnow, D. van Melkebeek & N. Vinodchandran (2006). Computational depth: concept and applications. *Theoretical Computer Science* **354**(3), 391–404. ISSN 0304-3975.

C. Bennett (1988). Logical depth and physical complexity. In *A half-century survey on The Universal Turing Machine*, 227–257. Oxford University Press, Inc., New York, NY, USA. ISBN 0-19-853741-7.

G. Chaitin (1966). On the Length of Programs for Computing Finite Binary Sequences. *Journal of ACM* **13**(4), 547–569. ISSN 0004-5411.

R. Impagliazzo & A. Wigderson (1996). P = BPP unless E has sub-exponential circuits: Derandomizing the XOR Lemma (Preliminary Version). In *Proceedings of the 29th ACM Symposium on Theory of Computing*, 220–229. ACM Press.

A. Klivans & D. van Melkebeek (2002). Graph Nonisomorphism Has Subexponential Size Proofs Unless the Polynomial-Time Hierarchy Collapses. *SIAM Journal on Computing* **31**(5), 1501–1526. ISSN 0097-5397.

A. Kolmogorov (1950). *Foundations of the Theory of Probability*. Chelsea Publishing.

A. Kolmogorov (1965). Three approaches to the quantitative definition of information. *Problems of Information Transmission* **1**(1), 1–7.

L. Levin (1973). Universal Search Problems. *Problems Information Transmission* **9**, 265–266.

M. Li & P. Vitányi (2008). *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Publishing Company, Incorporated. ISBN 0387339981, 9780387339986.

P. Miltersen (2001). Derandomizing complexity classes. *Handbook of Randomized Computing* .

N. Nisan & A. Wigderson (1994). Hardness vs. randomness. *Journal of Computer and System Sciences* **49**, 149–167.

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

Manuscript received 20 March 2009

Luís Antunes
Faculty of Sciences University of Porto
and Instituto de Telecomunicações

Current address of Luís Antunes:
Rua Campo Alegre n 1021/1055,
4169-007 Porto, Portugal
lfa@dcc.fc.up.pt
http://www.dcc.fc.up.pt/~lfa

Lance Fortnow
Northwestern University

Current address of Lance Fortnow:
Tech L359, 2145 Sheridan Road,
Evanston, IL 60208-3118, USA
fortnow@eecs.northwestern.edu
http://lance.fortnow.com

Alexandre Pinto
ISMAI - Instituto Superior da Maia and
CCTC - Centro de Ciências e Tecnologias de Computação, Universidade do Minho

Current address of Alexandre Pinto:
Av. Carlos Oliveira Campos - Castêlo da Maia
4475-690 Avioso S. Pedro, Portugal
ampinto@docentes.ismai.pt

André Souto
Faculty of Sciences University of Porto
and Instituto de Telecomunicações

Current address of André Souto :
Rua Campo Alegre n 1021/1055,
4169-007 Porto, Portugal
andresouto@dcc.fc.up.pt
http://www.dcc.fc.up.pt/~andresouto