Information and Computation 271 (2020) 104483

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Polylog depth, highness and lowness for E

Philippe Moser

Computer Science Department, National University of Ireland Maynooth, Maynooth, co. Kildare, Ireland

ARTICLE INFO

Article history:
Received 9 April 2018
Received in revised form 16 September 2019
Accepted 8 October 2019
Available online 16 October 2019

Keywords:
Algorithmic information theory
Kolmogorov complexity
Bennett logical depth

ABSTRACT

We study the relations between the notions of highness, lowness and logical depth in the setting of complexity theory. We introduce a new notion of polylog depth based on time bounded Kolmogorov complexity. We show polylog depth satisfies all basic logical depth properties, namely sets in P are not polylog deep, sets with (time bounded)-Kolmogorov complexity greater than polylog are not polylog deep, and only polylog deep sets can polynomially Turing compute a polylog deep set. We prove that if NP does not have p-measure zero, then NP contains polylog deep sets. We show that every high set for E contains a polylog deep set in its polynomial Turing degree, and that there exist Low(E, EXP) polylog deep sets. Keywords: algorithmic information theory; Kolmogorov complexity; Bennett logical depth.

1. Introduction

Bennett [1] introduced the notion of logical depth, as an attempt to characterise *useful* information (e.g. the halting problem) as opposed to non-useful information (e.g. a Martin-Löf random sequence). Interestingly, non-deep sequences (called shallow sequences) can range from trivially organised sequences such as 01010101..., to completely unorganised ones such as a random sequence. Logically deep sequences are somewhere between these two extremes.

A sequence is Bennett deep [1] if the difference of every computable approximation of the Kolmogorov complexity of its initial segments, and the actual Kolmogorov complexity, is unbounded. This difference is called the depth magnitude of the sequence [2].

Moser and Stephan [2] studied the differences in computational power of sequences of different depth magnitudes, within the context of computability theory. They related logical depth to standard computability notions (e.g. highness, diagonally-non-computability and lowness). Highness and lowness are important characterisations of the computational power of sets used in computability theory [3]. Informally a set is high (resp. low) if it is useful (resp. not useful) given as an oracle. Among others, they showed that a Turing degree is high iff it contains a deep set of large depth magnitude [2]. They found that not all deep sets need be high, by constructing a low deep set (subsequently refined by Downey, McInerney and Ng into a low r.e. deep set [4]).

In this paper, we revisit the results of Moser and Stephan [2] within the context of computational complexity theory. There are two main difficulties in trying to do so. Firstly the quest for the "right" polynomial version of depth has turned out to be elusive. There are several issues inherent to complexity theory that are hard to overcome, while trying to translate logical depth from computability to complexity theory. It is non-trivial to find a polynomial depth notion that simultaneously satisfies all natural properties of Bennett's original notion. Secondly, translating the results from [2] to complexity theory is not straightforward, because the main techniques in [2] do not scale down to complexity theory. Several computability

E-mail address: pmoser@cs.nuim.ie.

https://doi.org/10.1016/j.ic.2019.104483
0890-5401/

techniques used in [2] (e.g. the equivalence between highness and dominating functions, the notion of DNC degrees) have no natural equivalent concept in complexity theory. Therefore the proof of the main result in this paper (the existence of low deep sets) rely on different techniques from the ones used in [2].

As mentioned above, finding a good polynomial depth notion is not straightforward. There have been several attempts by various authors over the last decade [5–7]. The main difficulty is to achieve a notion that satisfies the natural properties of Bennett's logical depth (among others: easy and hard sets are not deep, non-deep sets can't compute deep ones, some deep set exists), while at the same time is powerful enough to prove interesting results. All notions so far have been a compromise between these two goals, i.e. relaxing the satisfaction of some of the natural properties in order for the notion to be useful. See [7] for a summary of previous polynomial depth notions, and the compromises made by each notion.

In this paper we study a polylog version of depth as close as possible to the original notion by Bennett, namely the difference of two Kolmogorov complexities with different time bounds. Informally Bennett's depth measures some aspect of the difference in power between Δ_2^0 and Δ_1^0. In our polylog setting, this becomes EXP vs. P. This corresponds to quasipolynomial vs. polylog in the setting of Kolmogorov complexity, because the size of the characteristic sequence is exponentially larger than the size of the strings it encodes (i.e. sets in P have their polynomial time complexity measured relative to the size of the input, which corresponds to polylog with respect to the size of the characteristic sequence). To allow for Kolmogorov complexity with polylog time bounds, i.e. such that there is not enough time to read the whole program, we use the oracle Kolmogorov complexity model of [8]. This model is equivalent to standard Kolmogorov complexity for time bounds linear or greater, but allows for sublinear time bounds.

Polylog depth is interesting because it interacts nicely with lowness and highness for the complexity class E, as explained below. But similarly to previous polynomial depth notions, this comes at a cost (the compromise): the natural property "random implies non-deep" is not formulated in terms of the canonical random sets for EXP (the ones on which no p_2-martingale succeeds), instead we use "complex" sets (sets for which the time bounded Kolmogorov complexity is large). It is currently unknown whether p_2-random sets and "complex" sets are the same, see [9] for more on this question. Apart from this compromise, polylog depth satisfies the natural properties of depth, namely "easy" sets (here: in P) and "random" enough sequences (here: complex sets) are shallow, and polylog depth satisfies a slow growth law, i.e. no shallow sequence can quickly compute a deep one. Also EXP contains polylog deep sets.

Next we investigate whether NP contains any polylog deep sets. Since one cannot exclude $\mathrm{P} = \mathrm{NP}$ (in which case all NP-sets are shallow), one can only hope for a conditional result. We prove that if NP does not have p-measure zero, then it contains a polylog deep set. The assumption NP does not have p-measure zero is a reasonable assumption based on Lutz's p-measure [10], which has implications not known to follow from $\mathrm{P} \neq \mathrm{NP}$. Examples include separating many-one and Turing reductions [11] and derandomization of AM [12,13], see [14] for more.

Next we study the relation between highness and depth, and show that each set that is high for E (i.e. a set A such that $\mathrm{E}^{\mathrm{E}} \subseteq \mathrm{E}^A$) contains a polylog deep set in its polynomial Turing degree. This is on par with the results of [2] at the polylog level. The idea of the proof is that highness enables the set to compute polynomially random strings of small sizes, but large enough to guarantee depth of the whole sequence.

Our main result investigates whether all polylog deep sets need be high. We find a negative answer, by constructing a polylog deep set in $\mathrm{Low}(\mathrm{E}, \mathrm{EXP})$, (i.e. sets A such that $\mathrm{E}^A \subseteq \mathrm{EXP}$). Lowness for E (i.e. $\mathrm{Low}(\mathrm{E}, \mathrm{E})$) was first studied in [15], where a low set in $\mathrm{E} - \mathrm{P}$ was constructed. The set constructed in [15] is a very sparse set of strings that are random at the polynomial time level but not at the exponential time level. The sparseness of the set guarantees that large queries can be answered with "no", hence only small queries need to be computed, which guarantees lowness. However the set in [15] is too sparse to be polylog deep. We construct a new set with blocks of subexponential size each containing a random string. The size of the blocks are measured by a tower of subexponential functions, in order to be able to satisfy two conflicting requirements, namely the sequence needs to be sparse enough to stay low, but the relative size of blocks needs to be small enough so that the polylog depth of each block is preserved over the whole sequence.

2. Preliminaries

Logarithms are taken in base 2 and rounded down. For simplicity of notation we write $\log n$ for $\lfloor \log_2 n \rfloor$ and $\log^{(2)} n$ stands for $\log\log n$. By convention, whenever a real number is cast to an integer, we take the floor of the real number; we omit the floor notation for simplicity of notation.

We use standard complexity/computability/algorithmic randomness theory notations (see [16,17,3,18,19]). We write $\{0,1\}^n$ for the set of strings of size n. We denote by $s_0, s_1, \ldots, s_n$ the standard enumeration of strings in lexicographic order. For a string x, its length is denoted by $|x|$. The empty string is $s_0 = \epsilon$. The index of string x is the integer $\mathrm{ind}(x)$ such that $x = s_{\mathrm{ind}(x)}$. For every natural number n, it holds $|s_n| = \log(n+1)$, and the index of strings of length n are in the interval $[2^n - 1, 2^{n+1} - 2]$. We identify n with s_n, in particular $|n| = |s_n| = \log(n+1)$. We say string y is a prefix of string x, denoted $y \prec x$, if there exists a string a such that $x = ya$.

A sequence is an infinite binary string, i.e. an element of 2^ω. For string or sequence S and $i, j \in \mathbb{N}$, we write $S[i, j]$ for the string consisting of the ith through jth bits of S, with the conventions that $S[i, j] = \epsilon$ if $i > j$, $S[i] = S[i, i]$, and $S[0]$ is the leftmost bit of S. We write $S \upharpoonright i$ for $S[0, i-1]$ (the first i bits of S) and $S \upharpoonright x$ for $S \upharpoonright \mathrm{ind}(x)$. The characteristic sequence of a set of strings L is the sequence $\chi_L \in 2^\omega$, whose nth bit is one iff $s_n \in L$. We abuse the notation and use L and χ_L interchangeably. Note that for any string x, $|L \upharpoonright x| = O(2^{|x|})$.

We consider the following standard time bound families: $\mathrm{P} = \{kn^k |\ k \in \mathbb{N}\}$, $\mathrm{PL} = \{k \log^k n |\ k \in \mathbb{N}\}$, $\mathrm{E} = \{2^{kn} |\ k \in \mathbb{N}\}$, and $\mathrm{EXP} = \{2^{n^k} |\ k \in \mathbb{N}\}$. We abuse notations by using time bound families for complexity classes interchangeably, e.g. $\mathrm{E} = \cup_{c \in \mathbb{N}} \mathrm{DTIME}(2^{cn})$. Logarithmic time bounds require oracle access to the input as explained below.

We use $\leq^+$ (resp. $=^+$) to denote less or equal (resp. equal) up to a constant term. We fix a poly-computable 1-1 pairing function $\langle \cdot \rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. An order function is an unbounded non-decreasing function from $\mathbb{N}$ to $\mathbb{N}$, computable in polynomial time, e.g. $\log n, n \log n, n^2$.

We consider standard polynomial Turing reductions $\leq_T^p$. Two sets A, B are polynomial Turing equivalent ($A \equiv_T^p B$) if $A \leq_T^p B$ and $B \leq_T^p A$. The polynomial Turing degree of a set A is the class of sets polynomial Turing equivalent to A.

Fix a universal Turing machine U. The (plain) Kolmogorov complexity of string x, denoted $C_U(x)$, is the length of the lexicographically first program x^* such that U on input x^* outputs x. It can be shown that the value of $C_U(x)$ does not depend on the choice of U up to an additive constant, therefore we drop U from the notation and write $C(x)$. $C(x, y)$ is the length of a shortest program that outputs the pair $\langle x, y \rangle$, and $C(x \mid y)$ is the length of a shortest program such that U outputs x when given y as an advice.

We also consider time bounded Kolmogorov complexity. To allow logarithmic time bounds i.e. shorter than the time required to read the full program, we use the oracle model of [8]. In this model, the universal machine is provided with program p as an oracle (written U^p), and can query any bit of it. As noticed in [8], the definition coincides with the standard time bounded Kolmogorov complexity, for time bounds greater than $O(n)$. Given time bound $t \geq \log n$, define

$$C^t(x) = \min\{|p| :\ \forall b \in \{0, 1, \epsilon\}\ \forall i \leq n, U^p(i, b)[t(n)] \downarrow = \text{accepts iff } x[i] = b\}$$

where $n = |x|$, $x[0], x[1], \ldots, x[n-1]$ are the n bits of x, $x[m] = \epsilon$ for all $m \geq n$ and $U^p(i, b)[t(n)] \downarrow$ means U with oracle access to p halts within $t(n)$ steps on input i, b.

To concatenate two programs p, q one can double all the bits in p, and append 01. We write $\hat{p}$ for this string, i.e. from $\hat{p}q$ it is easy to recover p, q.

A sequence A is called (PL, PL)-complex if for all $k, n \geq 0$, $C^{\log^k n}(A \upharpoonright n) >^+ \log^k n$. We use the same $(\cdot, \cdot)$ notation for other levels of complex sets, where the first (resp. second) parameter is the time bound (resp. the minimal program size bound).

The symmetry of information holds for exponential time bounds.

Theorem 2.1 *(Symmetry of information [20], 7.1.12, p. 548). Let $t \in \mathrm{E}$ be a time bound. Then there exists time bound $t' \in \mathrm{E}$ such that for any strings x, y, we have $C^t(x, y) \geq C^{t'}(x) + C^{t'}(y|x) - O(\log(|x| + |y|))$.*

Lutz [21] used Lebesgue measure to define a measure notion on complexity classes e.g. E, EXP.

For the rest of this section, let $i \in \{1, 2\}$; p_1 (abbreviated p) stands for polynomial time bounds, p_2 for 2^{PL}.

Definition 2.1 *(See [21] for more details).* A class C has p_i-measure zero (written $\mu_{p_i}(C) = 0$) if there is a single p_i-martingale d that succeeds on every language A of C.

This property is monotone in the following sense: If class D is contained in a class C of p_i-measure zero, then D also has p_i-measure zero. It is easy to see that if a class C has p_1-measure zero, then it has p_2-measure zero. The converse is not always true but the following is known.

Theorem 2.2 *(Juedes and Lutz [22]). NP has p_2-measure zero iff NP has p-measure zero.*

The measure conservation property [10] states that class E (resp. EXP) does not have p-measure (resp. p_2-measure) zero. Lutz [10] showed that sets with small circuit complexity have p_2-measure zero, which is implied by the following result.

Theorem 2.3 *(Lutz [10]). Let $c \in \mathbb{N}$. The set $\{A \in 2^\omega |\ \exists^\infty n\ C^n(A \upharpoonright n) < \log^c n\}$ has p_2-measure zero.*

3. Polylog depth

A sequence S is h-deep$_{\mathrm{PL}}$ if the difference between the large and small time-bounded Kolmogorov complexity of the prefixes of S exceeds some order function h. We call the order h the depth magnitude of S. As shown in [2], the choice of h can have consequences on the computational complexity of S.

Definition 3.1. Let $m(n) \leq n$ be an order function. A set S is m-deep$_{\mathrm{PL}}$ if for every $t \in \mathrm{PL}$ there exists $t' \in 2^{\mathrm{PL}}$ such that for almost every n, $C^t(S \upharpoonright n) - C^{t'}(S \upharpoonright n) \geq m(n)$.

We say S is $O(1)$-deep$_{\mathrm{PL}}$ if it is c-deep$_{\mathrm{PL}}$ for every $c \geq 0$. We say S is PL-deep$_{\mathrm{PL}}$ if it is m-deep$_{\mathrm{PL}}$ for every $m \in \mathrm{PL}$ (and similarly for other order function families).

In the following section, we show that polylog depth satisfies most natural properties (scaled down to complexity theory) of Bennett's original notion. Firstly, as noticed in [7], for most depth notions it can be shown that easy and random sequences are not deep. The following two results show that this is also the case for deep_{PL}.

Theorem 3.1. *Let A be in P. Then A is not $2\log n$-deep$_{PL}$.*

We need the following lemma; it follows the same proof as the unbounded Kolmogorov complexity case, but since we are considering sublinear time bounds, we need to check it still holds.

Lemma 3.1. *Let $A \in DTIME(n^c)$ (resp. $DTIME(2^{n^c})$). For every $t \in PL$ (resp. $t \in 2^{PL}$) and $n \in \mathbb{N}$, $C^{t'}(A \upharpoonright n) \leq^+ C^t(1^n)$ with $t'(n) = O(t(n) + \log^c n)$ (resp. $t'(n) = O(t(n) + 2^{\log^c n})$).*

Proof of Lemma 3.1. Let A, t be as above and let $n \in \mathbb{N}$ and M be a TM deciding A in time n^c. Let p denote a minimal t-program for 1^n, i.e. $|p| = C^t(1^n)$. Consider prefix-free program $p' = \hat{q}p$, where q are instructions such that for $b = 0, 1, \epsilon$ $U^{p'}(i, b)$ simulates $U^p(i, 1)$. If $U^p(i, 1)$ accepts (i.e. $i < n$), simulate $M(s_i)$ and accept iff $M(s_i) = b$. If $U^p(i, 1)$ rejects, then accept iff $b = \epsilon$. The first step takes at most $t(n)$ steps, and the simulation of M on an input of size at most $\log n$ takes at most $\log^c n$ steps, for a total of $O(t(n) + \log^c n)$ steps. The exponential case is similar. □

Proof of Theorem 3.1. Let us prove the theorem. Let $A \in \text{P}$, decidable in time n^k, let $t(n) = O(\log n + \log^k n)$ be given by Lemma 3.1, and let $t' \in 2^{\text{PL}}$. For every $n \in \mathbb{N}$,

$$\begin{aligned} C^t(A \upharpoonright n) - C^{t'}(A \upharpoonright n) &\leq^+ C^{2\log n}(1^n) && \text{by Lemma 3.1} \\ &\leq^+ \log n \end{aligned}$$

where the last inequality holds by a print program, i.e. a program $p = \hat{q}e$ where q are instructions, e is a binary encoding of n, such that for $b \in \{0, 1, \epsilon\}$, $U^{p'}(i, b)$ accepts iff ($i \leq n$ and $b = 1$) or ($i > n$ and $b = \epsilon$). Therefore A is not $2\log n$-deep_{PL}, which ends the proof. □

From the point of view of Bennett's logical depth, shallow sequences are those whose structure is either very organised, e.g. the sequence 1010101010..., or lack any organisation at all, e.g. a random sequence. For the latter case, Bennett [1] showed that Martin-Löf random sequences are not deep. A similar result holds in our setting:

Lemma 3.2. *Let A be $(2^{PL}, n - 2\log n)$-complex, then A is not $2\log n$-deep$_{PL}$.*

Proof. Let A be as above i.e., for every $k \in \mathbb{N}$, for almost every n

$$C^{2^{\log^k n}}(A \upharpoonright n) > n - 2\log n. \tag{1}$$

Since $C^{2\log n}(A \upharpoonright n) \leq^+ n$ via a "print" program, we have $C^{2\log n}(A \upharpoonright n) - C^{2^{\log^k n}}(A \upharpoonright n) \leq^+ n - (n - 2\log n)$, i.e. A is not $2\log n$-deep_{PL}. □

Note such sequences exist: every Martin-Löf sequence satisfies $C(A \upharpoonright n) \geq n - K(n) - O(1) > n - 2\log n$; and not all such sequences are Martin-Löf (e.g. see [18] section 8.4: a no gap theorem for 1-randomness, for more details).

Bennett showed that producing a logical deep sequence requires a complex and lengthy computation [1]. For example truth-table reductions are not capable of computing a Bennett deep sequence from the empty set. This is known as the slow growth law, which states that if some set A truth-table computes some deep set B, then A is deep. A similar result holds for deep_{PL}, once the power of the corresponding reductions is adapted to the polynomial world. Truth-table reductions are too powerful in the setting of complexity theory (they can compute every set in every complexity class included in Δ^0_1), thus they need to be replaced with polynomial time Turing reductions.

Theorem 3.2. *Let $A, B \in \text{EXP}$, $A \leq^p_T B$ in time $O(n^c)$, and A is $\alpha \log^{ck} n$-deep$_{PL}$ for some $k, \alpha > 0$, then B is $\beta \log^k n$-deep$_{PL}$ for some $\beta > 0$.*

Lemma 3.3. *Let $A \leq^p_T B$, where en^c is the running time of the reduction. For every $s' \in PL$ (with $s' = \log^k n$) and $n \in \mathbb{N}$ we have*

$$C^s(A \upharpoonright 2^{a \log^{1/c} n}) \leq^+ C^{s'}(B \upharpoonright n) + 2\log n$$

for some $a > 0$ and $s(n) = \frac{2}{a^{ck+c}} \log^{ck+c} n$.

Proof of Lemma 3.3. Let A, B, c, e, s' be as above, M be the TM computing the reduction, and $n \in \mathbb{N}$. Let p be a minimal s'-program for $B \upharpoonright n$. Consider the following program p' for $A \upharpoonright 2^{a \log^{1/c} n}$, where $p' = \hat{q}\hat{q}'p$, q is a set of instructions, q' encodes n (i.e. $|\hat{q}'| \leq^+ 2 \log n$), and where $U^{p'}(i, b)$ does the following:

- Recover n.
- If $i \geq 2^{a \log^{1/c} n}$, accept iff $b = \epsilon$.
- If $i < 2^{a \log^{1/c} n}$, simulate $M^B(s_i)$, answering each query s_j to B by simulating $U^p(j, 0)$. Accept iff $b = M^B(s_i)$.

For the last item above, notice that all queries are within $B \upharpoonright n$, because $A \upharpoonright 2^{a \log^{1/c} n}$ codes for strings of length at most $a \log^{1/c} n$. Therefore the largest query has size at most $e(a \log^{1/c} n)^c = ea^c \log n$, i.e. a string with index at most $2^{1+ea^c \log n} - 2 < 2^{2ea^c \log n} < n$, by choosing a small enough. The simulation of $M^B(s_i)$ takes at most $e|s_i|^c \leq \log n$ steps, and there are at most $\log n$ queries to $B \upharpoonright n$, each requiring at most $s'(n)$ steps, thus a total of $s'(n) \log n + \log n \leq 2s'(n) \log n$. Let us express this time bound as a function of the input size $m = 2^{a \log^{1/c} n}$. We have $\frac{1}{a^c} \log^c m = \log n$, i.e. $2^{\frac{1}{a^c} \log^c m} = n$ hence the total number of steps becomes

$$2s'(2^{\frac{1}{a^c} \log^c m}) \frac{1}{a^c} \log^c m = \frac{2}{a^{ck+c}} \log^{c+ck} m. \quad \square$$

Proof of Theorem 3.2. Let us prove the theorem. Let A, B, α, k be as above, and en^c be the running time of the $\leq_T^p$-reduction, and $\beta > 0$ to be determined later. By contradiction, suppose B is not $\beta \log^k n$-deep$_{\mathrm{PL}}$ i.e., there exists $t = \log^d n$ such that for every $t' \in 2^{\mathrm{PL}}$ there exists an infinite set N such that for every $n \in N$, $C^t(B \upharpoonright n) < \beta \log^k n + C^{t'}(B \upharpoonright n)$. Suppose B is decidable in time 2^{n^b} (for some $b > 0$). Consider $\bar{t}(n) = \frac{2}{a^{cd+c}} \log^{cd+c} n$, where a is given by Lemma 3.3, and $t'(n) = n + 2^{\log^b n}$, and let $n \in N$ where N is the infinite set of lengths testifying the non depth of B for this t'. We have

$$\begin{aligned}
C^{\bar{t}}(A \upharpoonright 2^{a \log^{1/c} n}) &\leq^+ C^t(B \upharpoonright n) + 2 \log n && \text{by Lemma 3.3} \\
&\leq \beta \log^k n + C^{t'}(B \upharpoonright n) + 2 \log n && \text{because } n \in N \\
&\leq^+ \beta \log^k n + C^n(1^n) + 2 \log n && \text{by Lemma 3.1} \\
&\leq^+ \beta \log^k n + \log n + 2 \log n \\
&< 4\beta \log^k n.
\end{aligned}$$

Thus for all $n \in N$ and all $s \in 2^{\mathrm{PL}}$,

$$C^{\bar{t}}(A \upharpoonright 2^{a \log^{1/c} n}) - C^s(A \upharpoonright 2^{a \log^{1/c} n}) \leq C^{\bar{t}}(A \upharpoonright 2^{a \log^{1/c} n}) - 0 < 4\beta \log^k n = \frac{4\beta}{a^{ck}} \log^{ck}(2^{a \log^{1/c} n})$$

i.e. A is not $\frac{4\beta}{a^{ck}} \log^{ck} n$-deep$_{\mathrm{PL}}$. Choosing $\beta > 0$ small enough such that $4\beta/a^{ck} < \alpha$, we get a contradiction. $\square$

The proof above implicitly shows the following: (PL, PL)-complex sets are closed upwards under poly Turing reductions. All PL-deep$_{\mathrm{PL}}$ sequences are (PL, PL)-complex. The converse holds if the sequence is in EXP. It would be interesting to see whether Theorem 3.2 still holds without the assumption that the set is in EXP, though this is currently unknown.

By enumerating all short $O(n)$-time programs, one can for every $k > 1$ construct a $(\mathrm{PL}, \log^{k+1} n)$-complex sequence in EXP, i.e. a sequence $A \in \mathrm{EXP}$ such that for all $n \in \mathbb{N}$, $C^n(A \upharpoonright n) > \log^{k+1} n$. Since $A \in \mathrm{EXP}$, by Lemma 3.1, A is $\log^k n$-deep$_{\mathrm{PL}}$.

It is natural to ask whether some NP sets are deep$_{\mathrm{PL}}$. One cannot exclude the possibility that P = NP, in which case the answer is negative, but if one assumes that NP is not a small subset of EXP, then one can show that NP contains deep sets. To measure the size of NP within EXP, we use Lutz's p-measure [10], which is a complexity version of Lebesgue measure that allows to measure the size of subsets of EXP. Thus the statement that NP is a not a small subset of EXP, is formalised by the statement that NP does not have p-measure zero. The assumption that NP does not have p-measure zero implies P≠NP and is not known to follow from it. It has been used to show some interesting results, e.g. [11–14], that are not known to follow from P≠NP.

Theorem 3.3. *If $C \subseteq \mathrm{EXP}$ with $\mu_{p_2}(C) \neq 0$ then C contains a $\log^k n$-deep$_{PL}$ set for every $k > 0$.*

Proof. Suppose $\mu_{p_2}(C) \neq 0$ and C does not contain a $\log^k n$-deep$_{\mathrm{PL}}$ set for some $k > 0$. Then for every $A \in C$ and for every $t' \in 2^{\mathrm{PL}}$ there exists an infinite set N, such that for every $n \in N$, $C^t(A \upharpoonright n) - C^{t'}(A \upharpoonright n) \leq \log^k n$. Thus $C^t(A \upharpoonright n) \leq \log^{k+1} n$, by Lemma 3.1, since $A \in \mathrm{EXP}$. Thus for every $A \in C$, there exists an infinite set N_A, such that for every $n \in N_A$, $C^n(A \upharpoonright n) < \log^{k+1} n$. Thus by Theorem 2.3, $\mu_{p_2}(C) = 0$; a contradiction. $\square$

Corollary 3.1. *If $\mu_p(NP) \neq 0$ then NP contains a $\log^k n$-deep$_{PL}$ set for every $k > 0$.*

Proof. Apply Theorem 3.3 together with Theorem 2.2. □

4. Highness and depth

Highness and lowness are important characterisations of the computational power of sets used in computability theory [3]. Informally a set is high (resp. low) if it is useful (resp. not useful) given as an oracle. The notions were generalised to considering two classes, with applications in randomness theory [18,19]. Complexity versions of these notions were developed for NP [16,17], and E [15].

Definition 4.1. Let $C \subseteq D$ be two complexity classes.

1. Set A is Low(C, D) if $C^A \subseteq D$.
2. Set A is High(C, D) if $C^A \supseteq D^D$.

Set A is low for E if it is Low(E, E). Set A is high for E if it is High(E, E).

Lowness and highness for E are preserved under polynomial Turing reductions.

Lemma 4.1. *Let A be high (resp. low) for* E. *Then all sets in the polynomial Turing degree of A are high (resp. low) for* E.

Proof. Let A be high for E, B be in the $\leq_T^p$-degree of A and $L \in \mathrm{E}^{\mathrm{E}}$. By highness for E of A, there is an oracle TM M^A that decides L in time 2^{an}. On an input x of size n, M makes at most 2^{an} queries to A, each of size less than 2^{an}. Since A, B are in the same $\leq_T^p$-degree, there is a machine N^B deciding A in time n^b. Thus each query q to A can be answered by N^B in $|q|^b \leq (2^{an})^b = 2^{abn}$ steps. Thus L can be decided in time $2^{abn}2^{an} \leq 2^{2abn}$ with oracle access to B, i.e. $L \in \mathrm{E}^B$. The proof for lowness is similar. □

The following result shows that any high set computes a deep set. The idea of the proof is that highness enables the set to compute polynomially random strings of small sizes, but large enough to guarantee depth of the whole sequence.

Theorem 4.1. *Let A be high for* E. *Then the $\leq_T^p$-degree of A contains a set that is* $\log^{(2)}$ *n-deep$_{PL}$.*

Proof. Let $c \in \mathbb{N}$ to be determined later. Consider the following set of random strings $R = \{x|\ C^{2^{2^{3n}}}(x) \geq |x|\} \in \mathrm{E}^{\mathrm{E}} \subseteq \mathrm{E}^A$. Let $j \in \mathbb{N}$, $j' = 2^{1+\log j^c}$ so that $j^c \leq j' \leq 2j^c$ and $\log j' = 1 + \log j^c$. Define

$$B[2^j - 1, 2^{j+1} - 2] = R[j' - 1, 2j' - 2]A[2^{j-1} - 1, 2^j - 2]0^{2^j - j' - 2^{j-1}}$$

i.e. $B \cap \{0,1\}^j$ codes for $R \cap \{0,1\}^{\log j'}$ and $A \cap \{0,1\}^{j-1}$.

Claim 1. *B is in the $\leq_T^p$-degree of A.*

For each x of length $\log j'$, deciding whether $x \in R$ requires at most $2^{a|x|}$ queries of size at most $2^{a|x|}$ to A (for some $a \in \mathbb{N}$), i.e. $2^{a \log j'} \leq O(j^{ac})$ queries of size $2^{a \log j'} = O(j^{ac})$. Since there are j' such x's, we have $R[j', 2j' - 2]$ can be computed in at most $O(j^{2ca})$ steps, hence $B \leq_T^p A$. Since $A \leq_T^p B$ the claim is proved.

Let us show that B is deep. Let $t \in \mathrm{PL}$, $j \in \mathbb{N}$ and $v \in [2^{j+1} - 1, 2^{j+2} - 2]$. Since this guarantees all of $R \cap 2^{\log j'}$ is available from $B \restriction v$, we have

$$C^t(B \restriction v) \geq C^n(B \restriction v) \geq^+ C^{2^n}(r, v) \tag{2}$$

where r is the first string of size $\log j'$ in R. Let $t_1 \in \mathrm{E}$ be the time bound given by Theorem 2.1 with $t_0(n) = 2^n$, so that we have

$$C^{2^n}(r, v) \geq C^{t_1}(r) + C^{t_1}(v \mid r) - O(\log |v|) \tag{3}$$

Let p be testifying $C^{t_1}(v \mid r)$, i.e. $U(p, r) = v$ in at most $t_1(|v|)$ steps. Let $p' = \hat{q}\hat{q}'p$ where q are instructions, q' is an encoding of j, and U on input p':

- Recovers j then j'.
- Computes r, i.e. the first element of $R[j' - 1, 2j' - 2]$.
- Simulates $U(p, r) = v$ (in $t_1(|v|)$ steps) and outputs v.

For the second item above: each bit of R requires at most $2^{\log j'}$ programs to be simulated for $2^{2^{3\log j'}}$ steps, i.e. a total of $j'2^{j'^3}$ steps, thus a total of at most $2^{j'^4} \leq 2^{j^{5c}}$ steps to compute $R[j'-1, 2j'-2]$. The total running time is less than $t_2(|v|) = t_1(|v|) + 2^{|v|^{5c}}$, since $|v| = 1 + j$. Therefore we have

$$C^{t_2}(v) \leq C^{t_1}(v \mid r) + |\hat{q}| + |\hat{q}'| \leq^+ C^{t_1}(v \mid r) + 2\log j. \tag{4}$$

Also $B \in \mathrm{EXP}$ because $A \in \mathrm{E}$ and $R[j'-1, 2j'-2]$ requires at most $2^{j^{5c}}$ steps to be computed, i.e. $B \in \mathrm{EXP}$.

Let p testify $C^{t_2}(v)$, i.e. $U(p) = v$ in $t_2(|v|)$ steps. Let $p' = \hat{q}p$ where U on input p' simulates $U(p) = v$ (in $t_2(|v|)$ steps), and outputs $B \upharpoonright v$ (in $v2^{|v|^b}$ steps, where $B \in \mathrm{Dtime}(2^{n^b})$) with a total running time less than $t_3(v) = t_2(|v|) + v2^{|v|^b} \leq t_2(\log(v+1)) + v2^{\log^b(v+1)} \in 2^{\mathrm{PL}}$. Thus

$$C^{t_3}(B \upharpoonright v) \leq^+ C^{t_2}(v). \tag{5}$$

We have,

$$\begin{aligned}
&C^t(B \upharpoonright v) - C^{t_3}(B \upharpoonright v) \geq^+ \\
&C^{t_1}(r) + C^{t_1}(v \mid r) - O(\log|v|) - C^{t_2}(v) \geq && \text{by Equations (3), and (5)} \\
&C^{t_1}(r) + C^{t_2}(v) - 2\log j - C^{t_2}(v) - O(\log|v|) \geq && \text{by Equation (4)} \\
&C^{t_1}(r) - O(\log j)
\end{aligned}$$

Since $t_1 \in \mathrm{E}$, we have $C^{t_1}(r) \geq |r| = \log j' \geq \log j^c$ because $r \in R$. Thus,

$$C^t(B \upharpoonright v) - C^{t_3}(B \upharpoonright v) \geq c\log j - O(\log j) \geq 2\log j \geq \log^{(2)} v$$

for an appropriate choice of c. Since $t \in \mathrm{PL}$ and $v \in \mathbb{N}$ are arbitrary, and $t_3 \in 2^{\mathrm{PL}}$, B is $(\log^{(2)} n)$-deep$_{\mathrm{PL}}$. □

5. Lowness and depth

The following result shows that some deep sets can be low. The idea of the proof is to construct a very sparse set of strings that are random at the polynomial time level but not at the exponential time level. The sparseness of the set guarantees that large queries can be answered with "no", hence only small queries need be computed, which guarantees lowness. To make the set deep, one needs to cut the sequence in blocks of subexponential size each containing a random string. The blocks need be large enough to not hurt the sparseness property, but small enough to ensure that the depth of the blocks is preserved over the whole sequence.

Theorem 5.1. *For every $\epsilon > 0$, there exists a set in* Low(E, EXP) *which is* $\log^{1/(1+\epsilon)} n$*-deep*$_{PL}$.

Proof. Let $\epsilon > 0$, and $\epsilon' = \epsilon/2$. For all $n \geq 1$, define $T_{n+1} = 2^{2^{\log^2 T_n}}$ with $T_0 = 1$. Consider the following set A where $A[T_n, T_{n+1}-1]$ is constructed as follows: For all $k \leq \frac{4}{\log(1+\epsilon')}\log^{(2)} T_n$ with $2^{\log^{(1+\epsilon')^{\frac{k+1}{2}}} T_n} < T_{n+1}$, put the lex-first string x_k^n of $R = \{x : C^{2^{n^2}}(x) \geq |x|\}$ with $2^{\log^{(1+\epsilon')^{\frac{k}{2}}} T_n} \leq \mathrm{ind}(x_k^n) < 2^{\log^{(1+\epsilon')^{\frac{k+1}{2}}} T_n}$ into A. Note such a string exists since $\forall n\ R \cap \{0,1\}^n \neq \emptyset$, and all the strings of length $1 + \log^{(1+\epsilon')^{\frac{k}{2}}} T_n$ are included in this interval: strings of length m have their index within $[2^m - 1, 2^{m+1} - 2]$. Since $\log^{(1+\epsilon')^{\frac{k+1}{2}}} T_n = (\log^{(1+\epsilon')^{\frac{k}{2}}} T_n)^{(1+\epsilon')^{1/2}}$, the right endpoint of the interval is large enough, provided T_n is large, because $(1+\epsilon')^{1/2} > 1$. Also add the lex-first string of R of length $\log(T_{n+1}) - 1$. Note all strings of this length have their index within interval $[T_n, T_{n+1} - 1]$.

Claim 2. $A \in \mathrm{EXP}$.

This holds because to decide whether $x \in R$ one needs to simulate $2^{|x|}$ programs for $2^{|x|^2}$ steps i.e. less than $t_A(|x|) = 2^{2|x|^2}$ steps.

Claim 3. *A is* Low(E, EXP).

Let us prove the claim. Let $L \in \mathrm{E}^A$; we need to show that $L \in \mathrm{EXP}$. Let M^A be an oracle TM deciding L in time 2^{cn}. Let $n \in \mathbb{N}$ and $v \in [T_n, T_{n+1} - 1]$. $M^A(v)$ makes at most $2^{c|v|}$ queries of size at most $2^{c|v|}$ to A. First note that all queries are within $A \upharpoonright T_{n+2}$. Indeed since $|v| \leq \log T_{n+1}$, the largest query to A has size at most $2^{c\log T_{n+1}} = T_{n+1}^c$, i.e. a string with index less than $2^{1+T_{n+1}^c} < 2^{2^{\log^2 T_{n+1}}} = T_{n+2}$. Let us show the queries to A can be answered in exponential time.

Subclaim 1. *Let q be a query to A by $M^A(v)$ with $|q| \geq |v|^4$. Then $C^{2cn}(q) < |q|$.*

Let us prove the claim. Any such q can be specified by its index in $M^A(v)$'s queries list (i.e. $O(c|v|)$ bits) with the index of the previous queries to A to be answered with yes (the other ones are answered with no), i.e. at most $O(|v| \log^2 T_n) < |v|^4$ bits because the max number of 1s (i.e. the number of yes queries) in $A[T_0, T_{n+2} - 1]$ is less than

$$\begin{aligned} a \log^{(2)} T_{n+1} + a \log^{(2)} T_n + \ldots + a \log^{(2)} T_1 &\leq a \log^{(2)} T_{n+1} + an \log^{(2)} T_n \\ &\leq (a+1) \log^{(2)} T_{n+1} \\ &= (a+1) \log^2 T_n \\ &\leq (a+1)|v|^2, \end{aligned}$$

because $|v| \geq \log T_n$, and where $a = \frac{4}{\log 1+\epsilon'}$. It takes at most $2^{c|v|}$ steps to simulate $M^A(v)$, plus $|v|^4$ steps to check the list of yes query answers, i.e. a total of less than $2^{2c|v|}$. Thus $C^{2cn}(q) < |v|^4 \leq |q|$ which proves the claim. Consequently, if $|q| \geq |v|^4$, then $A(q) = 0$ since $R(q) = 0$. Hence only queries with $|q| < |v|^4$ need be computed to simulate $M^A(v)$. Such queries to A are answerable in $2^{2|q|^2} \leq 2^{2|v|^8}$ steps, because $A \in \mathrm{Dtime}(2^{2n^2})$. Thus $M^A(v)$ can be simulated in $2^{O(|v|^8)}$ steps, which proves A is in Low(E, EXP).

Claim 4. *A is $\log^{1/(1+\epsilon)} n$-deep$_{PL}$.*

Let $t \in \mathrm{PL}, n \geq 1$, and $v \in [T_n, T_{n+1} - 1]$. Let $0 < k < \frac{4}{\log(1+\epsilon')} \log^{(2)} T_n$ be such that $v \in [2^{\log^{(1+\epsilon')} \frac{k}{2} T_n}, 2^{\log^{(1+\epsilon')} \frac{k+1}{2} T_n} - 1]$ ($k = 0$ will be done later). Note that such a k exists because when $k = \frac{4}{\log(1+\epsilon')} \log^{(2)} T_n$, then $2^{\log^{(1+\epsilon')} \frac{k+1}{2} T_n} > T_{n+1}$ (applying log on both sides of the equation three times).

We have

$$C^t(A \upharpoonright v) \geq C^n(A \upharpoonright v) \geq^+ C^{2^n}(x^n_{k-1}, v)$$

because given $A \upharpoonright v$ one can find v, and find x^n_{k-1} (corresponding to the last bit equal to 1 in $A \upharpoonright 2^{\log^{(1+\epsilon')} \frac{k}{2} T_n} \prec A \upharpoonright v$) in $v \leq 2^{|v|}$ steps. Let $t_1 \in \mathrm{E}$ be the time bound given by choosing $t_0(n) = 2^n$ in Theorem 2.1. By Theorem 2.1 we have

$$C^t(A \upharpoonright v) \geq^+ C^{2^n}(x^n_{k-1}, v) \geq C^{t_1}(x^n_{k-1}) + C^{t_1}(v \mid x^n_{k-1}) - O(\log |v|). \tag{6}$$

Let p be a program testifying $C^{t_1}(v \mid x^n_{k-1})$, i.e. $U(p, x^n_{k-1}) = v$ in $t_1(|v|)$ steps. Let $p' = \hat{q}\hat{q}'p$ where q are instructions, q' encodes k, n, ϵ', and such that $U(p')$

- recovers k, n, ϵ'
- finds x^n_{k-1} i.e. the first string in R with $2^{\log^{(1+\epsilon')} \frac{k-1}{2} T_n} \leq \mathrm{ind}(x^n_{k-1}) < 2^{\log^{(1+\epsilon')} \frac{k}{2} T_n}$
- simulates $U(p, x^n_{k-1}) = v$
- outputs v.

For each bit of $R \upharpoonright x^n_{k-1} + 1$, there are less than $2^{|x^n_{k-1}|}$ programs to simulate, each for at most $2^{|x^n_{k-1}|^2}$ steps, i.e. at most $2^{2|x^n_{k-1}|^2} \leq 2^{2|v|^2}$ steps. The simulation of $U(p, x^n_{k-1}) = v$ takes less than $t_1(|v|)$ steps, thus a total of $t_2(|v|) = t_1(|v|) + 2^{2|v|^2}$ steps. Since $|\hat{q}'| \leq^+ 2 \log k + 2 \log n$, we have

$$C^{t_2}(v) \leq^+ C^{t_1}(v \mid x^n_{k-1}) + 2 \log k + 2 \log n < C^{t_1}(v \mid x^n_{k-1}) + \log |v|.$$

Let $p' = \hat{q}p$ where and p is a minimal program testifying $C^{t_2}(v)$, and q are instructions. U on input p' simulates $U(p) = v$, and outputs $A \upharpoonright v$. The simulation of $U(p)$ takes at most $t_2(|v|)$ steps. The computation of the first v bits on A takes time $v 2^{2|s_v|^2} = v 2^{2 \log^2(v+1)} \leq 2^{\log^3 v}$ (because $A \in \mathrm{Dtime}(2^{2n^2})$), i.e. a total of less than $2^{\log^3 v} + t_2(|v|) = 2^{\log^3 v} + t_2(\log(v+1))$. Letting $t_3(n) = 2^{\log^3 n} + t_2(\log(n+1)) \in 2^{\mathrm{PL}}$, we have $C^{t_2}(v) \geq^+ C^{t_3}(A \upharpoonright v)$, hence

$$C^{t_3}(A \upharpoonright v) \leq^+ C^{t_2}(v) \leq C^{t_1}(v \mid x^n_{k-1}) + O(\log |v|). \tag{7}$$

We have,

$$\begin{aligned} &C^t(A \upharpoonright v) - C^{t_3}(A \upharpoonright v) \geq \\ &C^{t_1}(x^n_{k-1}) + C^{t_1}(v \mid x^n_{k-1}) - C^{t_1}(v \mid x^n_{k-1}) - O(\log |v|) \geq \qquad \text{by Equations (6) and (7)} \end{aligned}$$

$$|x^n_{k-1}| - O(\log|v|) \geq \log^{\frac{1}{1+\epsilon}} v \qquad \text{because } x^n_{k-1} \in R$$

because $|x^n_{k-1}| \geq \log^{(1+\epsilon')^{\frac{k-1}{2}}} T_n$, thus $|x^n_{k-1}|^{(1+\epsilon')} \geq \log^{(1+\epsilon')^{\frac{k+1}{2}}} T_n > \log v$, thus $|x^n_{k-1}| - O(\log|v|) > \log^{\frac{1}{1+\epsilon'}} v - O(\log|v|) = \log^{\frac{1}{1+\epsilon'}} v - O(\log^{(2)} v) > \frac{1}{2}\log^{\frac{1}{1+\epsilon'}} v > \log^{\frac{1}{1+\epsilon}} v$, for large enough v and because $\epsilon' = \epsilon/2$.

The case $k = 0$ is similar: the same proof applies except that x^n_{k-1} is replaced with the lex-last string x whose index bit is 1 in $A[T_{n-1}, T_n - 1]$. By construction of A the length of x is $|x| = \log(T_n) - 1$. Since $k = 0$, $\log v < \log^{(1+\epsilon')^{1/2}} T_n$, and similarly to the previous argument, we have

$$\begin{aligned} C^t(A \upharpoonright v) - C^{t_3}(A \upharpoonright v) &\geq |x| - O(\log|v|) \\ &\geq \log(T_n) - O(\log^{(2)} v) \\ &\geq \log^{\frac{1}{(1+\epsilon')^{1/2}}} v - O(\log^{(2)} v) \\ &> \log^{\frac{1}{1+\epsilon'}} v \\ &> \log^{\frac{1}{1+\epsilon}} v \end{aligned}$$

Since $v \in \mathbb{N}, t \in \mathrm{PL}$ are arbitrary and $t_3 \in 2^{\mathrm{PL}}$, B is $\log^{1/(1+\epsilon)} n$-deep$_{\mathrm{PL}}$. □

6. Discussion

The slow growth law (Theorem 3.2) holds when both sets are in EXP. It would be interesting to see whether this is required.

Although EXP contains a $\log n$-deep$_{\mathrm{PL}}$ set, the bounds in the slow growth law (Theorem 3.2) are not tight enough to prove that every EXP-complete set is $\log n$-deep$_{\mathrm{PL}}$. It would be interesting to show that this holds (or not). Highness and lowness have also been studied within the polynomial time hierarchy, i.e. for the class NP. Our polylog depth notion can be adapted to yield a depth measure between P and NP; however, it is not obvious how to translate our highness and lowness results in that setting. Highness/lowness for E is based on running time functions dominating each other (similarly to the domination properties of high sets in computability theory). On the other hand, highness for NP is based on nested levels of nondeterministic computations (similarly to the arithmetical hierarchy in computability theory).

The techniques in Theorem 5.1 fall short of constructing a polylog deep set that is low for E. It would be interesting to see if such a set exists.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

We thank all the anonymous referees for useful comments on earlier versions of this paper.

References

[1] C.H. Bennett, Logical depth and physical complexity, in: The Universal Turing Machine, A Half-Century Survey, 1988, pp. 227–257.
[2] P. Moser, F. Stephan, Depth, highness and DNR degrees, Discret. Math. Theor. Comput. Sci. 19 (4) (2017), http://dmtcs.episciences.org/4012.
[3] P. Odifreddi, Classical Recursion Theory: The Theory of Functions and Sets of Natural Numbers, Vol. 1, Springer, 1992.
[4] R. Downey, M. McInerney, K.M. Ng, Lowness and logical depth, Theor. Comput. Sci. 702 (2017) 23–33, https://doi.org/10.1016/j.tcs.2017.08.010.
[5] L. Antunes, L. Fortnow, D. van Melkebeek, N. Vinodchandran, Computational depth: concept and applications, Theor. Comput. Sci. 354 (2006) 391–404.
[6] D. Doty, P. Moser, Feasible depth, in: S.B. Cooper, B. Löwe, A. Sorbi (Eds.), CiE, in: Lecture Notes in Computer Science, vol. 4497, Springer, 2007, pp. 228–237.
[7] P. Moser, On the polynomial depth of various sets of random strings, Theor. Comput. Sci. 477 (2013) 96–108, https://doi.org/10.1016/j.tcs.2012.10.045.
[8] E. Allender, H. Buhrman, M. Koucký, D. van Melkebeek, D. Ronneburger, Power from random strings, SIAM J. Comput. 35 (6) (2006) 1467–1493.
[9] J.M. Hitchcock, N.V. Vinodchandran, Dimension, entropy rates, and compression, J. Comput. Syst. Sci. 72 (4) (2006) 760–782, https://doi.org/10.1016/j.jcss.2005.10.002.
[10] J. Lutz, Almost everywhere high nonuniform complexity, J. Comput. Syst. Sci. 44 (1992) 220–258.
[11] J.H. Lutz, E. Mayordomo, Cook versus Karp-Levin: separating completeness notions if NP is not small, Theor. Comput. Sci. 164 (1&2) (1996) 141–163, https://doi.org/10.1016/0304-3975(95)00189-1.
[12] V. Arvind, J. Köbler, On resource-bounded measure and pseudorandomness, in: S. Ramesh, G. Sivakumar (Eds.), Foundations of Software Technology and Theoretical Computer Science, 17th Conference, Kharagpur, India, December 18–20, 1997, Proceedings, in: Lecture Notes in Computer Science, vol. 1346, Springer, 1997, pp. 235–249.

[13] R. Impagliazzo, P. Moser, A zero-one law for RP and derandomization of AM if NP is not small, Inf. Comput. 207 (7) (2009) 787–792.
[14] J.H. Lutz, The quantitative structure of exponential time, in: L.A. Hemaspaandra, A.L. Selman (Eds.), Complexity Theory Retrospective II, Springer-Verlag, 1997, pp. 225–254.
[15] R.V. Book, P. Orponen, D.A. Russo, O. Watanabe, Lowness properties of sets in the exponential-time hierarchy, SIAM J. Comput. 17 (3) (1988) 504–516, https://doi.org/10.1137/0217030.
[16] J.L. Balcazar, J. Diaz, J. Gabarro, Structural Complexity I, EATCS Monographs on Theoretical Computer Science, vol. 11, Springer Verlag, 1995.
[17] J.L. Balcazar, J. Diaz, J. Gabarro, Structural Complexity II, EATCS Monographs on Theoretical Computer Science, vol. 22, Springer Verlag, 1990.
[18] R.G. Downey, D.R. Hirschfeldt, Algorithmic Randomness and Complexity, Springer, 2010.
[19] A. Nies, Computability and Randomness, Oxford University Press, 2009.
[20] M. Li, P. Vitanyi, Introduction to Kolmogorov Complexity and Its Applications, Springer, 1993.
[21] J. Lutz, The quantitative structure of exponential time, in: L. Hemaspaandra, A. Selman (Eds.), Complexity Theory Retrospective II, Springer, 1997, pp. 225–260.
[22] D.W. Juedes, J.H. Lutz, Weak completeness in E and E2, Theor. Comput. Sci. 143 (1) (1995) 149–158, https://doi.org/10.1016/0304-3975(95)80030-D.