Avoiding Simplicity is Complex[*]

Eric Allender

Department of Computer Science

Rutgers University

Piscataway, NJ 08855, USA

allender@cs.rutgers.edu

Holger Spakowski

Department of Mathematics & Applied Mathematics

University of Cape Town

Rondebosch 7701, South Africa

Holger.Spakowski@uct.ac.za

October 29, 2010

Abstract

It is a trivial observation that every decidable set has strings of length n with Kolmogorov complexity $\log n + O(1)$ if it has any strings of length n at all. Things become much more interesting when one asks whether a similar property holds when one considers *resource-bounded* Kolmogorov complexity. This is the question considered here: Can a feasible set A avoid accepting strings of low resource-bounded Kolmogorov complexity, while still accepting some (or many) strings of length n?

More specifically, this paper deals with two notions of resource-bounded Kolmogorov complexity: Kt and KNt. The measure Kt was defined by Levin more than three decades ago and has been studied extensively since then. The measure KNt is a nondeterministic analog of Kt. For all strings x, $\text{Kt}(x) \geq \text{KNt}(x)$; the two measures are polynomially related if and only if $\text{NEXP} \subseteq \text{EXP/poly}$ [6].

Many longstanding open questions in complexity theory boil down to the question of whether there are sets in P that avoid all strings of low Kt complexity. For example, the EXP vs ZPP question is equivalent to (one version of) the question of whether avoiding simple strings is difficult: (EXP = ZPP if and only if there exist $\epsilon > 0$ and a "dense" set in P having no strings x with $\text{Kt}(x) \leq |x|^\epsilon$ [5]).

Surprisingly, we are able to show *unconditionally* that avoiding simple strings (in the sense of KNt complexity) is difficult. Every dense set in NP $\cap$ coNP contains infinitely many strings x such that $\text{KNt}(x) \leq |x|^\epsilon$ for every $\epsilon > 0$. The proof does not relativize. As an application, we are able to show that if E = NE, then accepting paths for nondeterministic exponential time machines can be found

---

[*] A preliminary version of this work appeared in Proc. Computability in Europe (CiE) 2010 [4].

1

somewhat more quickly than the brute-force upper bound, if there are many accepting paths.

Key Words: Hitting Sets, Kolmogorov Complexity, Complexity Theory

1   Introduction

It has been observed before that many popular conjectures in complexity theory can be restated equivalently in terms of questions about the resource-bounded Kolmogorov complexity of feasible sets, and that this restatement can serve to highlight some of the tension among these conjectures. For instance, it is common to conjecture that

1. The containment $\mathrm{NTime}(t(n)) \subseteq \mathrm{DTime}(2^{O(t(n))})$ is nearly optimal, and that

2. Cryptographically-secure one-way functions exist.

The first of these two conjectures implies that there are polynomial time-bounded Turing machines that, for infinitely many input lengths n, accept some strings in Σ^n, but none having Kt-complexity less than (say) $n/5$ [2], where Kt is a time-bounded version of Kolmogorov complexity defined by Levin [18]. (Definitions will be provided in Section 2.) In contrast, the second conjecture implies that secure pseudorandom generators exist [14], which in turn implies that any polynomial time-bounded machine *must* accept some strings in Σ^n with Kt-complexity much less than $\sqrt{n}$ *if* the machine accepts at least half of the strings in Σ^n [1]. Thus, if the popular conjectures are true, sets in P *can* avoid accepting strings with low Kt-complexity, but *only* if they don't accept many strings of any given length. If a set in P contains a lot of strings of a given input length, then it *cannot* avoid accepting some simple strings, according to the popular belief.

This paper deals with the question of how difficult it is to avoid simple strings (i.e., strings of low resource-bounded Kolmogorov complexity) while still accepting a large number of strings. The main contribution of the paper is to present one setting in which we can replace popular conjecture and vague belief with unconditional theorems, showing that easy-to-compute sets *must* contain simple strings if they contain many strings. We also present an application of this new insight to the question of whether accepting computation paths of nondeterministic exponential time machines are easy to find, assuming "only" E = NE.

Let us introduce some notation, to help us gauge how successfully a set is avoiding simple strings. For any set $A \subseteq \Sigma^*$, define $\mathrm{Kt}_A(n)$ to be $\min\{\mathrm{Kt}(x) : x \in A^{=n}\}$, where $A^{=n} = A \cap \Sigma^n$. (For a definition of Levin's measure $\mathrm{Kt}(x)$, see Section 2.) If $A^{=n} = \emptyset$, then $\mathrm{Kt}_A(n)$ is undefined. Note that the rate of growth of $\mathrm{Kt}_A(n)$ is a measure of how successfully A avoids strings of low Kt complexity. The rate of growth of $\mathrm{Kt}_A(n)$ for sets A in P and P/poly is especially of interest, as can be seen from the following theorem. (We give a more precise definition of "dense" in Section 3, but for now it is sufficient to consider a set to be "dense" if it contains at least $2^n/n$ strings of each length n. An "NE search problem" is the task of mapping an input x to an accepting computation of M on input x, if one exists, where M is an NE machine.)

Theorem 1   *1. There is an* NE *search problem that is not solvable in time* $2^{2^{o(n)}}$ *if and only if there is a set* $A \in$ P *and an* $\epsilon > 0$ *such that* $\mathrm{Kt}_A(n) \neq O(n^\epsilon)$ *[2, Theorem 6]. (The set A need not be dense.)*

2. *There is an* NE *search problem that is not solvable in time* $2^{O(n)}$ *if and only if there is a set* $A \in$ P *such that* $\mathrm{Kt}_A(n) \neq O(\log n)$ *[2, Theorem 6]. (The set A need not be dense.)*

3. EXP $\not\subseteq$ P/poly *if and only if for every dense set* $A \in$ P/poly *and every* $\epsilon > 0$, $\mathrm{Kt}_A(n) = O(n^\epsilon)$ *[3, Theorem 12].*

4. *There is a set* $B \in$ E *and an* $\epsilon > 0$ *such that, for all large n, there is no circuit of size* $2^{\epsilon n}$ *accepting* $B^{=n}$ *if and only if for every dense set* $A \in$ P/poly, $\mathrm{Kt}_A(n) = O(\log n)$ *[3, Theorem 13].*

5. EXP $\neq$ ZPP *if and only if for every set* $A \in$ P *of polynomial density* [1] *and every* $\epsilon > 0$, $\mathrm{Kt}_A(n) = O(n^\epsilon)$ *[5].*

A nondeterministic analog of Levin's Kt measure, denoted KNt, was introduced recently [6]. For any set A, let $\mathrm{KNt}_A(n)$ be $\min\{\mathrm{KNt}(x) : x \in A^{=n}\}$. The rate of growth of $\mathrm{KNt}_A(n)$ is similarly related to open questions in complexity theory:

Theorem 2 *[6, Theorem 44] There is a set* $B \in$ NE/lin *such that for all large n, there is no nondeterministic circuit of size* $2^{\epsilon n}$ *accepting* $B^{=n}$ *if and only if for every dense set A in* NP/poly $\cap$ coNP/poly, $\mathrm{KNt}_A(n) = O(\log n)$.

Theorem 2 presents a condition regarding $\mathrm{KNt}_A(n)$ for dense sets A in a *nonuniform* class, and Theorem 1 contains analogous conditions regarding dense sets in both uniform *and* nonuniform classes. It is natural to wonder if Theorem 2 can be extended, to say something about the corresponding uniform class, and the experience of Theorems 1 and 2 could lead one to expect that such an extension would consist of showing that a statement about the KNt-complexity of dense sets in NP $\cap$ coNP is equivalent to some longstanding open question in complexity theory.

Thus it is of interest that our main theorem shows *unconditionally* that $\mathrm{KNt}_A(n)$ grows slowly for all dense sets in NP $\cap$ coNP.

The rest of the paper is organized as follows. Definitions and preliminaries are presented in Section 2. The main results are presented in Section 3. An application to NE search problems is presented in Section 4.

2   Preliminaries

We assume that the reader is familiar with complexity classes such as P, ZPP, NP, AM, and PSPACE; for background consult a standard text such as [7]. We use the following

---

[1] As discussed in Section 3, there is a slight difference between a "dense set" and a "set of polynomial density" which we prefer to ignore in this introduction. For the other parts of this theorem, the stated equivalence holds both for "dense sets" and "sets of polynomial density"; however it is not known if "set of polynomial density" can be replaced by "dense set" here.

notation for deterministic and nondeterministic exponential-time complexity classes: $E = \text{DTime}(2^{O(n)})$, $\text{NE} = \text{NTime}(2^{O(n)})$, $\text{EXP} = \text{DTime}(2^{n^{O(1)}})$, and $\text{NEXP} = \text{NTime}(2^{n^{O(1)}})$. $\text{P}^{\text{NP}[n]}$ is the class of languages accepted by polynomial-time oracle Turing machines with an oracle from NP, where the oracle machine makes at most n oracle queries on inputs of length n.

For any complexity class $\mathcal{C}$, and function $h(n) : \mathbb{N} \to \mathbb{N}$, let $\mathcal{C}/h(n)$ denote the class of sets B such that, for some "advice function" $a(n) : \mathbb{N} \to \Sigma^{h(n)}$, and some set $A \in \mathcal{C}$, $x \in B$ if and only if $(x, a(|x|)) \in A$. $\mathcal{C}/\text{poly}$ denotes $\bigcup_k \mathcal{C}/n^k + k$; $\mathcal{C}/\text{lin}$ denotes $\bigcup_k \mathcal{C}/kn$. The class P/poly has an equivalent characterization as the class of problems solvable by families of polynomial-size circuits. Note in particular that $(\text{NP} \cap \text{coNP})/\text{poly}$ is quite possibly a proper subset of $\text{NP}/\text{poly} \cap \text{coNP}/\text{poly}$.

Levin defined $\text{Kt}(x)$ to be $\min\{|d| + \log t : U(d) = x$ in time $t\}$ [18], where U is some fixed universal Turing machine. (It is important to note that Levin's definition is *independent* of any run-time t; the "t" that appears in the definition is a quantity that participates in the minimization expression.) Later, it was observed that $\text{Kt}(x)$ is polynomially related to the oracle circuit size that is required to compute the function that has x as its truth table [6], where the oracle is a complete set for E. In order to obtain a time-bounded notion of Kolmogorov complexity in the spirit of Levin's Kt function that is related to circuit complexity for more general oracles (including the empty oracle), a new measure, called KT, was defined [5]:

Definition 1 *Let U be a universal Turing machine and let B be an oracle. Define the measure $\text{KT}^B(x)$ to be*

$$\text{KT}^B(x) = \min\{\, |d| + t \ : \ U^{B,d} \text{ describes } x \text{ in time } t, \ (\text{meaning that}$$
$$\forall b \in \{0, 1, *\} \, \forall i \leq |x| + 1, U^{B,d}(i, b) \text{ accepts in } t \text{ steps iff } x_i = b)\}.$$

(The notation "$U^{B,d}(i, b)$" indicates that the machine U has random access (or "oracle access") to both the string d and the oracle B. This allows the running time to be less than $|d|$.) We omit the superscript B if $B = \emptyset$.

It is known that one can pick a complete set C for E such that Levin's definition of $\text{Kt}(x)$ is linearly-related to $\text{KT}^C(x)$ [5].

A nondeterministic analog of Kt called KNt was recently investigated [6], and it was shown that $\text{KNt}(x)$ is linearly related to $\text{KT}^D(x)$ for a set D that is complete for NE. Thus, for this paper, we will let $\text{Kt}(x)$ and $\text{KNt}(x)$ denote $\text{KT}^C(x)$ and $\text{KT}^D(x)$ for this E-complete set C and NE-complete set D, respectively.

For a given set A, and oracle B, we define $\text{KT}^B_A(n)$ to be equal to $\min\{\text{KT}^B(x) : x \in A^{=n}\}$. Thus $\text{Kt}_A(n) = \text{KT}^C_A(n)$, and $\text{KNt}_A(n) = \text{KT}^D_A(n)$.

We assume that the reader is familiar with polynomial-time Turing reducibility, denoted $\leq^{\text{p}}_T$. We also need to make use of reductions computed by polynomial-size *circuits*, instead of polynomial-time *machines*. A P/poly-Turing reduction of a set A to a set B is a family of polynomial-size circuits computing A, where the circuits have *oracle gates* for B, in addition to the usual AND and OR gates. (An oracle gate for B outputs 1 if the string y that is presented to it as input is an element of B.) If a P/poly-Turing reduction has the property that there is no path in the circuit from one oracle gate to another, then it is called a P/poly-truth-table reduction, denoted $\leq^{\text{P}/\text{poly}}_{tt}$.

3   Main Result

The main theorem applies only to languages that have "sufficiently many" strings; we call such sets "dense". The following definition makes precise exactly what sort of "density" is required:

Definition 2 *A set $A \subseteq \{0,1\}^*$ is* dense *if there is a k such that for every n there is some m with $n \leq m \leq n^k + k$ such that $|A \cap \{0,1\}^m| \geq 2^m / m^k$.*

Theorem 3 *Let A be a dense set in $\mathrm{NP} \cap \mathrm{coNP}$. Then for every $\epsilon > 0$ there are infinitely many $x \in A$ such that $\mathrm{KNt}(x) < |x|^\epsilon$.*

Proof: Most of the work has already been done in an earlier paper, in which it was shown that R_{KNt} (the set of "KNt-random strings", i.e., the set of strings x such that $\mathrm{KNt}(x) \geq |x|$) is not in $\mathrm{NP} \cap \mathrm{coNP}$ [6]. It was noticed only shortly after that paper was submitted for publication that the lower bound applied not only to R_{KNt}, but in fact to *every* dense set A such that, for some $\epsilon > 0$, $\mathrm{KNt}_A(n) = \Omega(n^\epsilon)$. We need to recall some of the main theorems of earlier work on this topic.

One of the main insights obtained in earlier work is that for "large" complexity classes, dense sets having only strings of high Kolmogorov complexity are hard under P/poly reductions. The following definition captures the property that a "large" class needs to have, in order for the proof to go through:

Definition 3 *A set B is* PSPACE-*robust if* $\mathrm{P}^B = \mathrm{PSPACE}^B$.

The notion of PSPACE-robustness was defined by Babai et al. [9], who observed that every set that is complete for EXP under $\leq_T^{\mathrm{p}}$ reductions is PSPACE-robust. Later, it was shown that NEXP also has this property [6].

Theorem 4 *[5, Theorem 31] Let B be any* PSPACE-*robust set. Let A be a set such that for some $\epsilon > 0$ and k, for every n there is some m such that*

- $n \leq m \leq n^k + k$,
- $|A \cap \{0,1\}^m| \geq 2^m / m^k$
- $\mathrm{KT}_A^B(m) \geq m^\epsilon$.

Then B is reducible to A via $\leq_{tt}^{\mathrm{P/poly}}$ reductions.

(This is a slight modification of the statement of the theorem as given in [5]. There, it was assumed that A contains many strings of *every* length, and contains *no* strings of low KT^B complexity. However, the $\leq_{tt}^{\mathrm{P/poly}}$ reduction that is given in [5] has the property that, on inputs of length n, all queries to the oracle A have the same length m, and the reduction works properly as long as, for the given length m, A contains many strings and no strings of low KT^B complexity. Thus, by simply encoding the length m into the nonuniform advice of the $\leq_{tt}^{\mathrm{P/poly}}$ reduction, the proof given in [5] suffices to establish Theorem 4.)

Since we are using the definition of KNt as KT^D for some set D that is complete for NE, and since every set that is complete for NE is PSPACE-robust, Theorem 4 immediately yields the following corollary:

Corollary 5 *Let A be any dense set such that $\mathrm{KNt}_A(n) = \Omega(n^\epsilon)$ for some $\epsilon > 0$. Then A is hard for* NEXP *under* $\leq_{tt}^{\mathrm{P/poly}}$ *reductions.*

We also need to use the result that any A that satisfies the hypothesis of Corollary 5 is also hard for PSPACE under probabilistic reductions:

Theorem 6 *[5, Theorem 33 and Lemma 35] Let A be any set of polynomial density, such that $\mathrm{Kt}_A(n) = \Omega(n^\epsilon)$ for some $\epsilon > 0$. Then* PSPACE $\subseteq$ ZPP^A.

Note that the term "polynomial density" as used in [5] is slightly more restrictive than the term "dense" as defined in this paper, since a set has "polynomial density" if it contains many strings of *every* length.

Corollary 7 *Let A be any dense set in* NP $\cap$ coNP *such that $\mathrm{KNt}_A(n) = \Omega(n^\epsilon)$ for some $\epsilon > 0$. Then* PSPACE $\subseteq$ (NP $\cap$ coNP)$/O(\log n)$.

Proof: Note that $\mathrm{Kt}_A(n) \geq \mathrm{KNt}_A(n) = \Omega(n^\epsilon)$. Thus we would like to modify the proof of Theorem 6 to (nearly) obtain that PSPACE $\subseteq$ ZPP^A.

The difficulty is that in Corollary 7, we have a weaker notion of density than in Theorem 6. The proof of Theorem 6 given in [5] presents a ZPP reduction with the property that, on inputs of length n, there are lengths m_1 and m_2 such that all queries to the oracle have length either m_1 or m_2. (Queries to length m_1 are used to obtain a string of high complexity, which is then used in conjunction with the Impagliazzo-Wigderson construction [17] to derandomize a BPP^A reduction, which only makes queries of length m_2.) The length m_2 can be replaced by any m_2' such that $m_2 \leq m_2' \leq m_2^{O(1)}$, as long as the reduction is given suitable advice, saying which length m_2' has sufficiently many strings, and the length m_1 can be replaced by any m_1' at most polynomially larger than m_2'. Thus the ZPP reduction running in time n^k can be simulated by a (ZPP$^{\mathrm{NP} \cap \mathrm{coNP}})/c \log n$ computation, where c depends on k and on the density parameter of A. The corollary follows by observing that ZPP$^{\mathrm{NP} \cap \mathrm{coNP}} =$ NP $\cap$ coNP.

Although the informal presentation given in the preceding paragraph is essentially correct, one does need to be a bit careful, in order to make sure that one gives an algorithim that can be implemented in NP $\cap$ coNP, which works correctly if given the correct advice, instead of merely giving an NP/poly $\cap$ coNP/poly algorithm (i.e., an algorithm that exhibits NP $\cap$ coNP-like behavior *only* if given the proper advice). Thus we present a few more details.

Our NP $\cap$ coNP algorithm takes a tuple (x, m_2) as input, and first guesses a possible value for m_1 (among polynomially-many possibilities, chosen to be sufficiently large in relation to m_2) and guesses a string z in A of length m_2. Because A is dense, there will always be (several) computations that succeed in finding a string $z \in A$; any computation path that fails to find such a string will simply reject. Because $\mathrm{KNt}_A(n) = \Omega(n^\epsilon)$, the string z can be viewed as the truth table of a function that requires exponentially-large oracle circuits, for any oracle from NE [6]. Thus, in particular, it requires exponentially large oracle circuits when the oracle is A, and hence it can be used as the "hard function" in applying the Impagliazzo-Wigderson generator [17], to derandomize a BPP^A computation (as in the proof of Theorem 6 given in

[5]). It is important to note that *every* string in A has high complexity, and thus every computation that carries out a derandomized simulation of the BPP^A computation will agree on whether to accept or reject.[2] This part of the computation is completely deterministic, except for querying the oracle A. Since A is in $\text{NP} \cap \text{coNP}$, the answers to each query can be verified (and any computation path that fails to verify a query answer can simply reject). Our NP (coNP) algorithm will simply accept (reject) if this deterministic part of the simulation accepts. This shows that we have an algorithm in $\text{NP} \cap \text{coNP}$; this algorithm can be trusted to give the correct answer about input x if it is given the proper length m_2 as advice, giving a length where A contains many strings. $\square$

We now proceed with the proof of Theorem 3. The proof is by contradiction: Assume that A is a dense set in $\text{NP} \cap \text{coNP}$ such that, for all large $x \in A$, we have $\text{KNt}(x) \geq |x|^\epsilon$. That is, $\text{KNt}_A(n) = \Omega(n^\epsilon)$.

By Corollaries 7 and 5 we have $\text{PSPACE} \subseteq (\text{NP} \cap \text{coNP})/O(\log n)$ and $\text{NEXP} \subseteq \text{P}^A/\text{poly} \subseteq \text{P}^{(\text{NP} \cap \text{coNP})/O(\log n)}/\text{poly} = (\text{NP} \cap \text{coNP})/\text{poly}$.

It is known that if $\text{NEXP} \subseteq (\text{NP} \cap \text{coNP})/\text{poly}$ then $\text{NEXP} = \text{AM}$ [6, Theorem 29]. Thus under our assumptions we have

$$\text{NEXP} = \text{AM} = \text{PSPACE} \subseteq (\text{NP} \cap \text{coNP})/O(\log n).$$

This is a contradiction, since $(\text{NP} \cap \text{coNP})/O(\log n) \subseteq \text{P}^{\text{NP}[n]}/n$, and it was shown by Buhrman, Fortnow, and Santhanam [11] that NEXP is not contained in $\text{P}^{\text{NP}[n]}/n$. $\square$

It would be nice to know if a better upper bound on the KNt-complexity of dense sets in $\text{NP} \cap \text{coNP}$ (or in P) can be proved.

3.1 Does This Relativize?

The proof of Theorem 3 does not relativize, since it relies on Theorem 6 (which in turn relies on the characterization of PSPACE in terms of interactive proofs [19, 21]) and also Theorem 28 of [6] (which relies on the characterization of NEXP in terms of interactive proofs [8]). However, we do not know if the statement of Theorem 3 actually fails relative to some oracle.

In this section, we present an oracle relative to which there is a set $A \in \text{P}$ such that (a) for infinitely many lengths n, A contains at least half of the strings of length n, and (b) A contains no strings of KNt complexity less than n^ϵ (even for ϵ very close to 1). The set A fails to be "dense", because if A contains any strings of length n, then for all m such that $n < m \leq 2^{n^\epsilon}$, $A^{=m} = \emptyset$. Thus this fails to be an oracle relative to which the claim of Theorem 3 fails.

We find it convenient to formulate our proof in terms of nondeterministic *distinguishing* complexity, various versions of which have been studied in different contexts (see, e.g. [10, 6]). The following definition is from [6].

---

[2]An earlier version of this paper [4] claimed that the conclusion of Theorem 3 holds for every dense set in $(\text{NP} \cap \text{coNP})/n^{o(1)}$, but the proof fails, because the condition that z have high KNt-complexity seems hard to guarantee in that setting.

Definition 4 *Let U be a nondeterministic Turing machine. Define* $\text{KNDt}_U(x)$ *to be* $\min\{|d| + \log t \; : \; \forall y \in \Sigma^{|x|} \; U^d(y)$ *runs in time t and accepts iff $x = y\}$.*

As usual, we select a fixed fast universal nondeterministic machine U, and define $\text{KNDt}(x)$ *to be* $\text{KNDt}_U(x)$*. Via standard arguments it follows that for all U', we have* $\text{KNDt}(x) \le \text{KNDt}_{U'}(x) + c$ *for some constant c.*

Theorem 8 *For each $\epsilon < 1$, there exists a set A such that*

- *for infinitely many n, $|A^{=n}| \ge 2^{n-1}$, and*

- *for all $x \in A$, $\text{KNDt}^A(x) \ge |x|^\epsilon$.*

Proof: This proof is based on proofs of results in [12] and [10].

We create A in stages. We start with $A := \emptyset$.

Stage s. We pick a large n such that adding strings of length n to A does not influence the construction in previous stages. Let $P = \Sigma^{\le n^\epsilon}$ (the set of "nondeterministic programs" of length $\le n^\epsilon$). We construct a set $B \subseteq \Sigma^n$ with $|B| \ge 2^{n-1}$ such that for each $d \in P$, one of the following is true:

- $U^{A \cup B, d}$ accepts no string in B, or

- $U^{A \cup B, d}$ accepts at least two different strings of length n.

This will ensure that for any $x \in B$, $\text{KNDt}^{A \cup B}(x) \ge n^\epsilon$. (We will clock all these computations such that they will reject if they take time more than 2^{n^ϵ}.)

Construction, phase 1

1. $H := \Sigma^n$

2. $\Delta := \emptyset$.

3. **while** there exist $p \in P - \Delta$ and $R \subseteq \Sigma^n$ such that $|R| \le 2^{5n^\epsilon}$ and $\forall X \subseteq H - R$, $X \cap L(U^{A \cup X, p}) = \emptyset$ **do**

4. $\quad H := H - R$

5. $\quad \Delta := \Delta \cup \{p\}$

6. **end while**

Observations about phase 1

After phase 1, the following are true:

- For every $p \in \Delta$, we have

$$\forall X \subseteq H, X \cap L(U^{A \cup X, p}) = \emptyset.$$

Throughout stage s, B will always be a subset of H. Hence we only need to take care of the programs $p \in P - \Delta$, i.e., we only have to ensure that for every $p \in P - \Delta$, $U^{A \cup B, p}$ accepts at least two different strings of length n.

- If B is any subset of H such that $|H - B| \leq 2^{5n^\epsilon}$, then for each $p \in P - \Delta$, we have:

$$\exists X \subseteq H, \exists y \in X \text{ s.t. } U^{A \cup X, p} \text{ accepts } y.$$

and in particular, for $B = H$:

$$\exists X \subseteq B, \exists y \in X \text{ s.t. } U^{A \cup X, p} \text{ accepts } y.$$

(This holds because otherwise the **while** loop would have continued, by adding p to Δ.)

- In phase 1, we start with $H := \Sigma^n$ and then remove no more than $|P| \cdot 2^{5n^\epsilon}$ strings from H. Thus after phase 1,

$$|H| \geq 2^n - |P|2^{5n^\epsilon} \geq 2^n - 2^{n^\epsilon+1} \cdot 2^{5n^\epsilon} = 2^n - 2^{6n^\epsilon+1}.$$

Construction, phase 2

Enumerate $P - \Delta$ by $p_1, p_2, \dots, p_\ell$ and let $v = 3 \cdot 2^{n^\epsilon} \cdot |P|$.

1. $B := H$

2. **for** $i := 1$ **to** ℓ **do**

3.   **for** $j := 1$ **to** v **do**

4.   Pick a minimal $X \subseteq B$ s.t. $U^{A \cup X, p_i}$ accepts some $y \in X$. Let ρ be an accepting path for such a string y.

5. $Q_{p_i,j} := \{\alpha \in \Sigma^n \mid \alpha \text{ is queried on } \rho\} \cup \{y\}$.

6. $Q^+_{p_i,j} := Q_{p_i,j} \cap X$.

7. $Q^-_{p_i,j} := Q_{p_i,j} - X$.

8. $y_{p_i,j} := y$.

9. $B := B - Q_{p_i,j}$.

10.   **end for**

11. **end for**

Observations about phase 2

- In phase 2, we remove no more than

$$|P|v2^{n^\epsilon} = |P| \cdot 3 \cdot 2^{n^\epsilon} \cdot |P| \cdot 2^{n^\epsilon}$$

strings from B (note that 2^{n^ϵ} is the bound for the length of ρ), which is smaller than 2^{5n^ϵ} for large n. By an observation made after phase 1, this implies that in line 4 it is always possible to pick the set X.

- As we remove no more than 2^{5n^ϵ} strings from B in phase 2, we have $|H| - |B| \leq 2^{5n^\epsilon}$, and therefore, $|B| \geq 2^n - 2^{6n^\epsilon + 1} - 2^{5n^\epsilon} \geq 2^{n-1}$ for large enough n. In the remainder of stage s, we will only add strings to B. Hence the final set B will also have at least 2^{n-1} strings.

- The sets $Q^+_{p_i,j}$ are all disjoint.

- The strings $y_{p_i,j}$ are all distinct.

- B is disjoint with each of the sets $Q_{p_i,j}$, and the sets $Q^+_{p_i,j}$, and $Q^-_{p_i,j}$ partition $Q_{p_i,j}$.

Construction, phase 3 (selection phase)

In this phase, we use the information obtained in phase 2 to finally get B such that for every $p \in P - \Delta$, $U^{A \cup B, p}$ accepts at least two different strings of length n. It is easy to see that $U^{A \cup Y, p_i}(y_{p_i,j})$ accepts for any Y with $Q^+_{p_i,j} \subseteq Y$ and $Q^-_{p_i,j} \cap Y = \emptyset$.

We will find for each p_i an index set $P_i \subseteq \{1, 2, \ldots, v\}$ with $|P_i| \geq 2$ such that the following is satisfied:

For all $i, i' \in \{1, 2, \ldots, \ell\}$ and all $j \in P_i$ and $j' \in P_{i'}$,

$$Q^+_{p_i,j} \cap Q^-_{p_{i'},j'} = \emptyset. \tag{1}$$

Now we define our final set B as

$$B := B \cup \bigcup_{i \in \{1, \ldots, \ell\}, j \in P_i} Q^+_{p_i,j}.$$

It is easy to see that for each $i \in \{1, \ldots, \ell\}$, $U^{A \cup B, p_i}(y_{p_i,j})$ accepts for each $j \in P_i$. Thus for each $i \in \{1, \ldots, \ell\}$, $U^{A \cup B, p_i}$ accepts at least two different strings of length n.

We set $A := A \cup B$, and go to stage $s + 1$.

It remains to describe how to find the index sets P_i.

1. For each $i \in \{1, \ldots, \ell\}$ and $j \in \{1, \ldots v\}$, set pair (p_i, j) unmarked.

2. **for** $i := \ell$ **to** 1 **do twice**

3.   Pick the largest j such that (p_i, j) is unmarked.

4.   Mark (p_i, j).

5.   Mark all $(p_{i'}, j')$ with $i' < i$ or ($i' = i$ and $j' < j$) satisfying $Q^+_{p_{i'},j'} \cap Q^-_{p_i,j} \neq \emptyset$.

6. **end for**

7. For each $i \in \{1, \ldots, \ell\}$, let $P_i := \{j \mid (p_i, j)$ is not marked $\}$.

Observations about phase 3

- The sets $Q^+_{p_i,j}$ are all disjoint and all sets $Q^-_{p_i,j}$ are of size at most 2^{n^ϵ} (because of the bound for the computation time). Hence in each run of the **for** loop, each of the two $Q^-_{p_i,j}$ can intersect with at most 2^{n^ϵ} different sets $Q^+_{p_{i'},j'}$, which means that in each run of the **for** loop, we mark no more than $2(2^{n^\epsilon} + 1)$ pairs. Thus altogether, in phase 3 we mark no more than $|P| \cdot 2(2^{n^\epsilon} + 1)$ pairs, which is smaller than $v - 2$ for large n. However, for each $i \in \{1, \dots, \ell\}$, we started with v different unmarked pairs $(p_i, *)$. Hence there is always an unmarked (p_i, j) in line 3, and for each $i \in \{1, \dots, \ell\}$, at least two unmarked pairs $(p_i, *)$ remain at the end. Thus for each $i \in \{1, \dots, \ell\}$, $|P_i| \geq 2$.

- Let $i, i' \in \{1, \dots, \ell\}$ and $j, j' \in \{1, \dots, v\}$ be arbitrary with $i' < i$ or $(i' = i$ and $j' < j)$. The construction in phase 2 already ensured that

$$Q^-_{p_{i'},j'} \cap Q^+_{p_i,j} = \emptyset.$$

  Phase 3 additionally ensures that for unmarked pairs, i.e., for all $j \in P_i$ and $j' \in P_{i'}$,

$$Q^+_{p_{i'},j'} \cap Q^-_{p_i,j} = \emptyset.$$

  This shows that the constructed index sets P_i satisfy (1).

$\square$

Corollary 9 *For each $\epsilon < 1$, there exists a set A such that*

- *for infinitely many n, $|A^{=n}| \geq 2^{n-1}$, and*

- *for all $x \in A$, $\mathrm{KNDt}^A(x) \geq |x|^\epsilon$.*

Proof: Follows easily since it is shown in [6, Thm. 58] that $\mathrm{KNDt}(x) = \mathrm{KNt}(x) + \Theta(\log|x|)$, and the proof relativizes. $\square$

4 An Application to Search Problems

One of the aspects of the theory of NP-completeness that makes it so widely applicable, is the fact that, for NP-complete problems, *search* is equivalent to *decision*. That is, the problem of *deciding* membership in an NP-complete set is polynomially-equivalent to the problem of *finding* a proof of membership. Hartmanis, Immerman, and Sewelson observed that the proof of equivalence that works for NP-complete problems breaks down for exponential-time computations, and they asked whether search and decision are also equivalent for NE-complete problems [13]. A partial answer was provided by Impagliazzo and Tardos [16], who presented an oracle relative to which E = NE but relative to which there exists a nondeterministic exponential-time machine M such that there is no function computable in exponential time that maps each input x accepted by M to an accepting computation of M on input x. An alternative oracle construction was subsequently given by Buhrman, Fortnow, and Laplante [10].

The trivial brute-force deterministic algorithm for finding accepting computations of NE machines takes doubly exponential time $2^{2^{O(n)}}$. No significantly better upper bound is known, even for the special case of finding accepting computations of probabilistic NE machines, that have *many* accepting computation paths if they have any at all. This has been the case, even under the assumption that E = NE.

As a consequence of the results of Section 3, we can now say something nontrivial about an upper bound on the complexity of finding accepting computations of NE machines if E = NE – at least for certain classes of NE machines. (Actually, it suffices to use the weaker assumption that NEXP $\subseteq$ EXP/poly.) Let ZPE be the exponential-time analog of the complexity class ZPP. That is, B is in ZPE if there are two nondeterministic Turing machines M_0 and M_1 running for time 2^{cn} for some c, where M_0 accepts $\overline{B}$ and M_1 accepts B, with the property that if $x \in B$, then for at least half of the strings r of length 2^{cn}, M_1 accepts x along the computation path given by r, and if $x \notin B$, then for at least half of the strings r M_0 accepts x along the computation path given by r. Thus, for every string x, either half of the strings r of length $2^{c|x|}$ are accepting computations of M_1, or half of the strings r are accepting computations of M_0. A ZPE *search problem* (defined by the machines M_0 and M_1) is the task of taking x as input, and producing a string r as output, that causes either M_0 or M_1 to accept.

Theorem 10 *If* NEXP $\subseteq$ EXP/poly*, then for every* ZPE *search problem, there is a deterministic algorithm M solving it with the property that, for every $\epsilon > 0$, M runs in time $2^{2^{\epsilon|x|}}$ for infinitely many x.*

Proof: Consider a ZPE search problem defined by machines M_0 and M_1. Let N be an NE machine running in time 2^{cn} that, on input x, guesses a string r of length 2^{cn} and accepts if r causes either M_0 or M_1 to accept on input x. (Note that N accepts *every* string x.)

Let $d : \mathbb{N} \rightarrow \{0,1\}^*$ be a standard bijection (e.g., $d(i)$ is the string x such that the binary representation of $i+1$ is $1x$). Let

$$W_N = \{r : \text{(some prefix of) } r \text{ causes } N$$
$$\text{to accept the string } d(n), \text{ where } n^{c+1} = |r|\}.$$

Note that, since $|d(n)| = O(\log n)$, W_N is in P, and W_N is dense (since it contains at least half of the strings of each length of the form n^{c+1}).

By Theorem 3, for every $\epsilon > 0$ there are infinitely many $r \in W_N$ such that $\text{KNt}(r) < |r|^\epsilon$. Since we are assuming that NEXP $\subseteq$ EXP/poly, it follows that Kt and KNt are polynomially related [6], and thus we have that for every $\epsilon > 0$ there are infinitely many $r \in W_N$ such that $\text{Kt}(r) < |r|^\epsilon$. Let C be the E-complete set such that $\text{Kt}(x) = \text{KT}^C(x)$.

Consider the following algorithm M: On input x, compute n so that $d(n) = x$. For $k = 1$ to n^c, for all descriptions d of length k, see if $U^{C,d}$ describes a string r of length n^c in time k. If so, and if r causes N to accept on input x, then halt and output r.

It is straightforward to verify that the algorithm M has the properties claimed for it in the statement of the theorem. $\square$

The conclusion of Theorem 10 holds for a great many more NE search problems than merely those in ZPE. It holds for any NE machine N for which the language W_N

constructed in the proof of Theorem 10 is dense. (This corresponds to those problems in NE that are accepted by NE machines that have many accepting computation paths for at least one string of every length (or, more generally, at least one string out of every $O(1)$ consecutive lengths).) Rather than creating a new definition to capture this class, we simply state the following corollary:

Corollary 11 *If* NEXP $\subseteq$ EXP/poly, *then for every* NE *search problem defined by an* NE *machine N such that the set W_N is dense, there is a deterministic algorithm M solving it with the property that, for every $\epsilon > 0$, M runs in time $2^{2^{\epsilon|x|}}$ for infinitely many x.*

As stated, Theorem 10 is actually quite a bit weaker than a result presented by Impagliazzo, Kabanets and Wigderson [15] where a stronger conclusion is shown to follow from a weaker hypothesis. (It does not appear that Corollary 11 is subsumed by [15].) More specifically, Impagliazzo, Kabanets, and Wigderson show that if EXP $\neq$ ZPP, then ZPE search problems can be solved not merely in time $2^{2^{\epsilon|x|}}$ for infinitely many x, but for *all* x of length n for infinitely many n [15, Theorem 47]. (The statement of their Theorem 47 does not explicitly give a running time for ZPE search problems, and instead is stated in terms of an upper bound for recognizing *languages* in ZPE, but their proof actually shows how to solve ZPE search problems.) Their conclusion is obviously stronger than the conclusion of Theorem 10; in order to see that the hypothesis of Theorem 10 is stronger, we need to show why NEXP $\subseteq$ EXP/poly implies ZPP $\neq$ EXP. If NEXP $\subseteq$ EXP/poly and ZPP $=$ EXP, then EXP $\subseteq$ P/poly, and thus NEXP $\subseteq$ P/poly, which implies NEXP $=$ MA [15, Theorem 23]. Thus NEXP $=$ EXP $=$ ZPP, which contradicts the nondeterministic time hierarchy theorem [20].

It is possible to give a more direct proof of this result of Impagliazzo, Kabanets, and Wigderson, by making use of part 5 of Theorem 1 (which was not available to them):

Theorem 12 *([15, Theorem 47]) If* EXP $\neq$ ZPP, *then, for every $\epsilon > 0$, ZPE search problems can be solved in time $2^{2^{\epsilon|x|}}$ for all x of length n, for infinitely many n.*

Proof: By Theorem 1, we know that if EXP $\neq$ ZPP, then for every set $A \in$ P of polynomial density, for every $\epsilon > 0$, $\text{Kt}_A(n) = O(n^\epsilon)$.

As in the proof of Theorem 10, consider a ZPE search problem defined by machines M_0 and M_1, and let N be an NE machine running in time 2^{cn} that, on input x, guesses a string r of length 2^{cn} and accepts if r causes either M_0 or M_1 to accept on input x.

Instead of the set W_N that is defined in the proof of Theorem 10, consider the set W_N': $\{r : 2^m 2^{cm} \leq |r| < 2^{m+1} 2^{c(m+1)}$, where $r = r_1 r_2 \ldots r_{2^m} z$ for some string z, and for each $i \leq 2^m$, r_i is a string of length 2^{cm} encoding an accepting computation of N on the ith string of length $m\}$. (A similar construction is employed by Impagliazzo, Kabanets, and Wigderson.) Note that W_N' is in P, and has polynomial density (since it contains at least half of the strings of each length n). Thus by our assumption, $\text{Kt}_{W_N'}(n) = O(n^\epsilon)$.

Now consider the following modification to the algorithm M from the proof of Theorem 10: On input x of length m, let x be the ith string of length m in lexicographical order. For $k = 1$ to n^c, for all descriptions d of length k, see if $U^{C,d}$ describes a string r of length between $2^m 2^{cm}$ and $2^{m+1} 2^{c(m+1)}$ in time k. If so, then let r_i be the substring of length 2^{cm} starting at position $(i-1)2^{cm}$. If r_i causes N to accept on input x, then halt and output r_i.

Note that for all $\epsilon > 0$, for infinitely many m we are guaranteed to find a description d of length at most $k = (2^m)^\epsilon$ such that $U^{C,d}$ describes a string $r \in W'_N$, which means that *every* substring r_i of r causes N to accept the ith string of length m. $\square$

5 Closing Comments

For sufficiently "powerful" forms of resource-bounded Kolmogorov complexity (such as KT^E where E is complete for EXPSPACE), the lexicographically first element of $A^{=n}$ will always have logarithmic complexity, for any $A \in \mathrm{P}$ [6]. Conceivably, one could define a version of resource-bounded Kolmogorov complexity related to a low level of the exponential-time hierarchy (with just a few alternations – and therefore conceptually "closer" to KNt than KT^E) where this same technique could be applied. It seems unlikely that KNt is powerful enough to always give logarithmic complexity to the lexicographically least element of $A^{=n}$, for every set A in P, although we know of no unlikely consequences, if that were to be the case.

Acknowledgments

References

[1] E. Allender. Some consequences of the existence of pseudorandom generators. *Journal of Computer and System Sciences*, 39:101–124, 1989.

[2] E. Allender. Applications of time-bounded Kolmogorov complexity in complexity theory. In *O. Watanabe (Ed.), Kolmogorov Complexity and Computational Complexity*, pages 4–22. Springer, 1992.

[3] E. Allender. When worlds collide: Derandomization, lower bounds, and Kolmogorov complexity. In *Proc. Conf. on Found. of Software Technology and Theo. Comp. Sci. (FST&TCS)*, volume 2245 of *Lecture Notes in Computer Science*, pages 1–15, 2001.

[4] E. Allender. Avoiding simplicity is complex. In *Programs, Proofs, Processes, Proc. 6th Conference of Computability in Europe, (CiE 2010)*, volume 6158 of *Lecture Notes in Computer Science*, pages 1–10, 2010.

[5] E. Allender, H. Buhrman, M. Koucký, D. van Melkebeek, and D. Ronneburger. Power from random strings. *SIAM Journal on Computing*, 35:1467–1493, 2006.

[6] E. Allender, M. Koucký, D. Ronneburger, and S. Roy. The pervasive reach of resource-bounded Kolmogorov complexity in computational complexity theory. *Journal of Computer and System Sciences*. To appear.

[7] Sanjeev Arora and Boaz Barak. *Computational Complexity, a modern approach*. Cambridge University Press, 2009.

[8] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[9] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993.

[10] H. Buhrman, L. Fortnow, and S. Laplante. Resource-bounded Kolmogorov complexity revisited. *SIAM Journal on Computing*, 31(3):887–905, 2002.

[11] Harry Buhrman, Lance Fortnow, and Rahul Santhanam. Unconditional lower bounds against advice. In *Proc. of International Conference on Automata, Languages, and Programming (ICALP)*, volume 5555 of *Lecture Notes in Computer Science*, pages 195–209, 2009.

[12] J. Goldsmith, L. A. Hemachandra, and K. Kunen. Polynomial-time compression. *Computational Complexity*, 2:18–39, 1992.

[13] Juris Hartmanis, Neil Immerman, and Vivian Sewelson. Sparse sets in NP-P: EXPTIME versus NEXPTIME. *Information and Control*, 65(2/3):158–181, 1985.

[14] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28:1364–1396, 1999.

[15] R. Impagliazzo, V. Kabanets, and A. Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *J. Comput. Syst. Sci.*, 65(4):672–694, 2002.

[16] R. Impagliazzo and G. Tardos. Decision versus search problems in super-polynomial time. In *Proc. IEEE Symp. on Found. of Comp. Sci. (FOCS)*, pages 222–227, 1989.

[17] R. Impagliazzo and A. Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proc. ACM Symp. on Theory of Computing (STOC) '97*, pages 220–229, 1997.

[18] L. A. Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61:15–37, 1984.

[19] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39:859–868, 1992.

[20] J. Seiferas, M. Fischer, and A. Meyer. Separating nondeterministic time complexity classes. *Journal of the ACM*, 25:146–167, 1978.

[21] A. Shamir. IP = PSPACE. *Journal of the ACM*, 39:869–877, 1992.