On Hard Instances

Martin Mundhenk [1]

Universität Trier, FB IV – Informatik, D-54286 Trier, Germany

Abstract

Instance complexity was introduced by Orponen, Ko, Schöning, and Watanabe [14] as a measure of the complexity of individual instances of a decision problem. Comparing instance complexity to Kolmogorov complexity, they introduced the notion of p-hard instances, and conjectured that every set not in P has p-hard instances. Whereas this conjecture is still unsettled, Fortnow and Kummer [6] proved that NP-hard sets have p-hard instances, unless P = NP. The unbounded version of the conjecture was proven wrong by Kummer [9].

We introduce a slightly weaker notion of hard instances. In the unbounded version, we characterize the classes of recursive enumerable resp. recursive sets by hard instances. In bounded versions, we characterize the class P. Hard instances are shown to be stronger than complexity cores (introduced by Lynch [11]). Nevertheless, NP-hard sets must have super-polynomially dense hard instances, unless P = NP.

1 Introduction

Orponen, Ko, Schöning, and Watanabe [14] introduced instance complexity as a measure of the complexity of individual instances of a decision problem. The t-time bounded instance complexity $\mathrm{ic}^t(x : A)$ of x w.r.t. A is the length of the shortest program which correctly computes $A(x)$ in time $t(|x|)$ and which also is consistent with A (i.e. on every input y it either outputs $A(y)$ in time $t(|y|)$ or "says" that it is unable to make a decision). In [14] it is shown that P is the class of sets with polynomial-time instance complexity bounded by a constant, formally $A \in \mathrm{P} \Leftrightarrow \exists$ polynomial p, constant $c\ \forall^\infty x : \mathrm{ic}^p(x : A) < c$. Because the complexity of each instance x is bounded by its Kolmogorov complexity

[1] A preliminary version was presented at MFCS'97 [13]. Supported in part by the Office of the Vice Chancellor for Research and Graduate Studies at the University of Kentucky, and by Deutsche Forschungsgemeinschaft (DFG), grant Mu 1226/2-1.

$\mathrm{C}(x)$, they conjectured the much stronger property that sets out of P must have infinitely many instances which reach this maximal complexity, even if we take a time bounded Kolmogorov complexity; i.e., for every set $A \notin \mathrm{P}$ and every polynomial q, there exists a polynomial q' and a constant c such that the set $\{x \mid \mathrm{ic}^q(x : A) > \mathrm{C}^{q'}(x) - c\}$ is infinite. (Note that $\mathrm{C}^{q'}(x)$ has no limit inferior, i.e. $\forall k \; \exists y \; \forall x \geqslant y : \; \mathrm{C}^{q'}(x) \geqslant k$.)

This *instance complexity conjecture* is still open. Buhrman and Orponen [5] settled the conjecture for E complete sets, Fortnow and Kummer [6] for all recursive tally sets and for recursive honest Turing NP-hard sets (unless P = NP in the latter case). The respective unbounded formulation for recursive sets holds for recursively enumerable complete sets [5], but does not hold in general as shown by Kummer [9].

In this paper, we start our investigations by considering how instance complexity and Kolmogorov complexity are related for recursively enumerable sets. We prove that each recursively enumerable set A has (unbounded) instance complexity bounded by the Kolmogorov complexity relative to the Halting Problem K_0 and A itself, on the instances which belong to A. Consequently, each recursive set A has (unbounded) instance complexity bounded by the Kolmogorov complexity relative to K_0 and A, i.e. A is recursive iff $\mathrm{ic}(x : A) \leqslant \mathrm{C}^{K_0 \oplus A}(x)$ for almost all x. Attaching polynomial time bounds and replacing K_0 by the NP-complete set SAT, we get an exact characterization of P, namely a set A is in P iff there exists a polynomial p such that for every polynomial p' and for almost every x, $\mathrm{ic}^p(x : A) \leqslant \mathrm{C}^{\mathrm{SAT} \oplus A, p'}(x)$. Compared to the characterization of P from [14] cited above, the right-hand side of our inequality has no limit inferior. Thus, it is stronger then their characterization. Nevertheless, it is still weaker than the instance complexity conjecture. For recursive sets A, we can omit the use of A as oracle and obtain $A \in \mathrm{P}$ iff $\exists \text{poly } p \; \forall \text{poly } p' \; \forall^\infty x : \; \mathrm{ic}^p(x : A) \leqslant \mathrm{C}^{\mathrm{SAT}, p'}(x)$. The only difference to the instance complexity conjecture is the use of the SAT oracle. When we omit this oracle, we are not able to keep the polynomial time bound and obtain $A \in \mathrm{P}$ iff $\exists \text{poly } p \; \forall^\infty x : \; \mathrm{ic}^p(x : A) \leqslant \mathrm{C}(x)$.

Orponen et al. [14] used their characterization of P to derive a notion of hardness for instances, which they show to be similar to that of complexity cores, as defined by Lynch [11]. Using our characterization of P, we get a different notion of hard instances for a set A, namely the set of hard instances $H(A) = \{x \mid \forall \text{ polynomial } p : \mathrm{ic}^p(x : A) > \mathrm{C}^A(x)\}$. It follows that every set not in P has an infinite set of hard instances. We show that our notion of hard instances is stronger than that of complexity cores. There are sets whose hard instances are sparse, whereas a complexity core has exponentially density. Moreover, a sparse set $H(A)$ of hard instances implies that A reduces to a sparse set via a composition of polynomial-time 2-truth-table and conjunctive reducibility. Consequently, no $\leq^p_{btt}$-hard set for NP has a sparse set of hard instances unless P = NP.

The paper is organized as follows. Section 2 gives basic notation and definitions, Section 3 gives characterizations of complexity classes in terms of instance complexity, and defines the notion of hard instances. Section 4 considers structural properties of hard instances.

2 Notation and Definitions

We consider strings over the alphabet $\Sigma = \{0,1\}$. The empty string is denoted ε. The length of a string x is denoted by $|x|$. Sets are considered to be subsets of Σ^*. A set A is called *tally*, if $A \subseteq 0^*$. A set is *sparse*, if its density is bounded by a polynomial. TALLY denotes the class of all tally sets, SPARSE that of sparse sets. $A \oplus B$ denotes the marked union of sets $\{0x \mid x \in A\} \cup \{1x \mid x \in B\}$.

Notions from structural complexity theory are defined in the standard manner (see e.g. [2,16]). Polynomial time reduction classes are denoted $R^p_\alpha(\mathcal{C}) = \{A \mid \exists B \in \mathcal{C} : A \leq^p_\alpha B\}$. The polynomial-time reducibilities used here are many-one $\leq^p_m$, conjunctive $\leq^p_c$, 2-truth table $\leq^p_{2\text{-}tt}$, bounded truth table $\leq^p_{btt}$, and Turing $\leq^p_T$.

Our model of computation is the deterministic (oracle) Turing machine. Each program $\pi \in \Sigma^*$ computes a partial function $\Sigma^* \to \Sigma^*$. For any string x, let $\pi(x)$ denote the output of π on input x.

We will give a short review of necessary definitions for Kolmogorov and instance complexity. We leave out details like the choice of a universal Turing machine with respect to which the size and the computation time of the programs are measured, and suppose that an "optimal interpreter" is used to run the programs. The existence, robustness, and invariance of such an interpreter and more details can be found e.g. in [7,14,10]. For simplicity and w.l.o.g. we assume that the program $\pi = \varepsilon$ denoted by the empty string halts on every input without any computation and output.

The *Kolmogorov complexity* of a string $x \in \Sigma^*$ relative to oracle A is

$$\mathrm{C}^A(x) = \min_{\text{program } \pi} \{|\pi| \mid \pi^A(\varepsilon) = x\} \quad .$$

The notion of time-bounded Kolmogorov complexity was introduced by Hartmanis [7]. The *t-time bounded Kolmogorov complexity* is

$$\mathrm{C}^{A,t}(x) = \min_{\text{program } \pi} \{|\pi| \mid \pi^A(\varepsilon) = x \text{ and } \pi^A(\varepsilon) \text{ makes at most } t(|x|) \text{ steps}\} \quad .$$

We use $\mathrm{C}^t(x)$ to denote $\mathrm{C}^{\emptyset,t}(x)$. Note that $\mathrm{C}^{A,t}$ (seen as function $\Sigma^* \to \mathbb{N}$) has neither a limit superior nor a limit inferior for every choice of A and t.

Proposition 1 (cf. [10, Theorem 2.5]) *Let A be any set and t be any function. Then*

$$\forall n \in \mathbb{N} \;\; \exists x_0 \in \Sigma^* \;\; \forall x \geqslant x_0 \;\; : \quad \mathrm{C}^{A,f}(x) \geq n.$$

The characteristic function of a set A is denoted by $A(\cdot)$, where $A(x) = 1$ if $x \in A$, and $A(x) = 0$ if $x \notin A$. For $a, b \in \{0, 1, \perp\}$, we denote $a \simeq b$, if either $a = b$ or $\perp \in \{a, b\}$. Let π be a Turing machine. Then $\pi(x) = \perp$ denotes that either π does not halt on input x, or it halts with output not in $\{0, 1\}$. Machine π is *consistent with A on input x* (denoted $\pi(x) \simeq A(x)$), if $\pi(x) = \perp$ or $\pi(x) = A(x)$. π is called *A-consistent* (denoted $\pi \simeq A$), if $\pi(x) \simeq A(x)$ for every $x \in \Sigma^*$.

The notion of (unbounded and bounded) instance complexity was introduced by Orponen, Ko, Schöning, and Watanabe [14] . The *instance complexity of a string x w.r.t. A* is defined as

$$\mathrm{ic}(x : A) = \min_{\text{program } \pi} \{|\pi| \mid \pi \simeq A \wedge \pi(x) = A(x)\} \quad .$$

For a function t, let $\pi(x)_t$ be the decision of π on input x after $t(|x|)$ steps, i.e. $\pi(x)_t = \pi(x)$ if π on input x halts after at most $t(|x|)$ steps with output 0 or 1, and $\pi(x)_t = \perp$ otherwise. π is called *t-time bounded A-consistent* (denoted $\pi_t \simeq A$), if $\pi(x)_t \simeq A(x)$ for every $x \in \Sigma^*$. Note that for $\pi = \varepsilon$, by the above convention it holds that $\pi_t \simeq A$ for every A and time bound t.

The *t-time bounded instance complexity of a string x w.r.t. A* is defined as

$$\mathrm{ic}^t(x : A) = \min_{\text{program } \pi} \{|\pi| \mid \pi_t \simeq A \wedge \pi(x)_t = A(x)\} \quad .$$

A set A is said to have *p-hard instances*, if for every polynomial q there exists a polynomial q' and infinitely many x such that $\mathrm{ic}^q(x : A) > \mathrm{C}^{q'}(x)$. This definition stems from [14], where it was conjectured that every set not in P has p-hard instances. This conjecture is known as the *Instance Complexity Conjecture.*

3 Hard Instances

The Instance Complexity Conjecture is known not to hold in the unbounded case. Kummer [9] showed that there exists a recursively enumerable but non-recursive set A such that $\mathrm{ic}(x : A) \leqslant \log \mathrm{C}(x)$ for almost every x. We show

```
input 0^n
Π := Σ^{≤n}              (* set of programs to be checked for A-consistency *)
for m := n, n+1, n+2, ... do
  for all π ∈ Π and all y ∈ Σ^m do
    if π(y) ≄ A(y) then Π := Π − {π} end
  end
  if ∃y ∈ A^{=m} ∀π ∈ Π : ¹  π(y) =⊥ then
    output the lexicographically first such y ∈ A^{=m} and halt
  end
end
```

Fig. 1. Algorithm N

---

how to change the conjecture to get an exact characterizations of the classes of recursive enumerable resp. of recursive sets. Essentially, the instance complexity has to be compared to the Kolmogorov complexity taken relative to the halting problem K_0 and to the considered set itself.

Theorem 2 *Let A be any subset of Σ^*. The following are equivalent.*

(1) A is recursively enumerable.
(2) For almost every $x \in A$: $\;\mathrm{ic}(x : A) \leqslant \mathrm{C}^{K_0 \oplus A}(x)$.

PROOF. If A is recursively enumerable, then there exists a Turing machine M accepting all instances in A. Thus for every $x \in A$: $\mathrm{ic}(x : A) \leqslant |M|$. By Proposition 1 it follows that $\mathrm{ic}(x : A) \leqslant \mathrm{C}^{K_0 \oplus A}(x)$ for almost every $x \in A$.

For the other proof direction, assume that A is not recursively enumerable. We show that algorithm N (see Figure 1) with the halting problem K_0 and A as oracles outputs hard instances.

Note that the outcome of $\pi(y)$ can be decided using the Halting Problem as oracle, and $A(y)$ resp. "$y \in A$" can be decided using oracle A. All other calculations of N are performed in finite time without use of an oracle. Therefore, every single step of the program can be executed in finite time using oracles K_0 and A.

Assume, that $N^{K_0 \oplus A}$ does not halt on some input 0^n. Since the outer for-loop must then be repeated for infinitely many m, the set Π cannot be changed from some m_0 on. Therefore, Π becomes a finite set of programs which are almost everywhere consistent with A, and almost every instance $x \in A$ is accepted by some program in Π (namely all instances of length at least m_0). Running all programs of Π in parallel and deciding like the first stopping one, yields a program with the above two properties of Π. Thus A is recursively enumerable, contradicting the above assumption. Therefore, $N^{K_0 \oplus A}$ must halt for every input 0^n.

Let y_n be the output of $N^{K_0 \oplus A}$ on input 0^n. Then $\{y_n \mid n \geqslant 0\}$ is an infinite set, because y_n exists and $|y_n| \geqslant n$. By construction of N, $y_n \in A$ and $\mathrm{ic}(y_n : A) > n$ for every n. On the other hand, each y_n is constructed by N on input 0^n. Therefore, there exists a constant c such that for every n, $\mathrm{C}^{K_0 \oplus A}(y_n) \leqslant \mathrm{C}^{K_0 \oplus A}(0^n) + c$, where c is essentially the size of N. Because for a constant c' and every n it holds that $\mathrm{C}^{K_0 \oplus A}(0^n) \leqslant \log n + c'$, and because $\log n + c + c' \leqslant n$ for almost every n, it follows that $\mathrm{ic}(y_n : A) > \mathrm{C}^{K_0 \oplus A}(y_n)$ for almost every n. Therefore, there exist infinitely many $x \in A$ such that $\mathrm{ic}(x : A) > \mathrm{C}^{K_0 \oplus A}(x)$. $\square$

Since recursive sets are exactly those which are recursively enumerable and have a recursively enumerable complement, we get the following characterization of recursive sets in terms of instance complexity.

Theorem 3 *Let A be any subset of Σ^*. The following are equivalent.*

(1) A is recursive.
(2) For almost every $x \in \Sigma^*$: $\mathrm{ic}(x : A) \leqslant \mathrm{C}^{K_0 \oplus A}(x)$.

The characterization of recursive sets from Theorem 3 can be turned into one for sets decidable within some time bound. In order to approach to the Instance Complexity Conjecture, we would like to bound the Kolmogorov complexity by some polynomial. As a matter of fact, we have to solve the search for a hard string in polynomial time, what is not known to be possible without the help of an NP oracle.

Theorem 4 *Let A be any subset of Σ^*. The following are equivalent.*

(1) $A \in \mathrm{P}$.
(2) There exists a polynomial p such that for every polynomial p' and almost every $x \in \Sigma^*$: $\mathrm{ic}^p(x : A) \leqslant \mathrm{C}^{\mathrm{SAT} \oplus A, p'}(x)$.

PROOF. If $A \in \mathrm{P}$, then there exist a polynomial p and a constant c such that $\mathrm{ic}^p(x : A) \leqslant c$ for almost every x [14]. For every polynomial p', there are only finitely many x with $c \leqslant \mathrm{C}^{\mathrm{SAT} \oplus A, p'}(x)$, proving the "forward" direction of the statement.

For the reverse proof direction, assume that $A \notin \mathrm{P}$. Fix any polynomial p. We modify the algorithm N from the proof of Theorem 2 to algorithm M given

[1] Note that Π does not become empty, since the program $\pi = \varepsilon$ (which halts undecided on every input) will not be removed from Π.

input 0^n
$\Pi := \Sigma^{\leqslant 3 \log n}$
for $m := n, n+1, n+2, \ldots$ **do** (* line 3 *)
  for all $\pi \in \Pi$ **do** (* line 4 *)
    if $\pi(s_m)_p \not\simeq A(s_m)$ **then** $\Pi := \Pi - \{\pi\}$ **end** (* line 5 *)
  end
  if $\exists y \in \Sigma^m \; \forall \pi \in \Pi : \pi(y)_p = \perp$ **then** (* line 7 *)
    output the lexicographically first such y and halt
  end
end

Fig. 2. Algorithm M

in Figure 2. There, s_i denotes the i-th string over Σ in the standard ordering. Note that M acts "delayed." Consider a repetition of the loop beginning at line 3. Whereas N removes from Π all programs which are not consistent with A on any string of length m, M removes from Π those programs which are not consistent with A on string s_m, which has length logarithmic in m. Nevertheless, as in the proof of Theorem 2, we can argue that M halts for every input 0^n.

We show that M with oracles SAT and A outputs hard instances. Let z_n be the output of M on input 0^n. We have to argue that M runs in time polynomial in $|z_n|$. First note that in every repetition of the loop beginning at line 3, for loop-counter m it holds that $|\Pi| \leqslant 2m^3 - 1$. Line 5 can be computed in time polynomial in m, say $s(m)$. Thus for any m, the execution of the complete loop beginning at line 4 takes $O(m^3 \cdot s(m))$ steps. Line 7 can be computed in polynomial time, say $s'(m)$, using a prefix search in the NP oracle $\{\langle \Pi, z, 1^m, 1^k \rangle \mid z \text{ is prefix of a string } y \in \Sigma^m \forall \pi \in \Pi : \pi(y)_k = \perp\}$. Therefore, the execution of one repetition of the loop beginning at line 3 for loop-counter m using SAT and A as oracles takes $O(r_0(m))$ steps for $r_0(m) = m^3 \cdot s(m) + s'(m)$ which can be chosen as nondecreasing polynomial. If eventually an output of length $|z_n|$ is produced and the algorithm halts, the running time is $p'(|z|) = O(|z| \cdot r_0(|z_n|))$.

Since $\mathrm{ic}^p(z_n : A) > 3 \log n$, and $2 \log n \geqslant \mathrm{C}^{\mathrm{SAT} \oplus A, p'}(z_n)$ for almost every n, the proof is completed. $\square$

If A is a recursive set, one can compute $A(s_j)$ without the help of oracle A. In order to modify M such that it is running in polynomial time independent on the time taken to decide $A(s_j)$, we bound the number of steps spend in the computation of $A(s_j)$. For α being a program deciding A, let line 5 in algorithm M be replaced by

if $\exists j \in \{0, \ldots, m\} : \; \pi(s_j)_p \not\simeq \alpha(s_j)_m$ **then** $\Pi := \Pi - \{\pi\}$ **end**

Each of the programs π in Π and α are run on $m+1$ inputs of length at most $1+\log m$. The running time of the π is bounded by polynomial p in the length of the input, and the running time of α is bounded by m. Therefore, this modified algorithm still has polynomial running time. Even though its computation is more "delayed," the same arguments as in the above proof hold. (Note that the $\log n$ gap between instance and Kolmogorov complexity of hard instances allows to increase the size of algorithm M by a constant $|\alpha|$.)

Theorem 5 *Let A be any recursive subset of Σ^*. The following are equivalent.*

(1) $A \in \mathrm{P}$.
(2) There exists a polynomial p such that for every polynomial p' and almost every $x \in \Sigma^*$: $\mathrm{ic}^p(x : A) \leqslant \mathrm{C}^{\mathrm{SAT},p'}(x)$.

If $\mathrm{P} = \mathrm{NP}$, each question to the SAT oracle can be answered in polynomial time. This yields another proof that the Instance Complexity Conjecture holds under this condition.

Theorem 6 [6, Theorem 19] *Assume that* $\mathrm{P} = \mathrm{NP}$*, and let A be any recursive subset of Σ^*. The following are equivalent.*

(1) $A \in \mathrm{P}$.
(2) There exists a polynomial p such that for every polynomial p' and almost every $x \in \Sigma^*$: $\mathrm{ic}^p(x : A) \leqslant \mathrm{C}^{p'}(x)$.

If we don't care about the time bounds for C, we can skip the NP oracle for the halting problem. This yields the following new characterization of P.

Theorem 7 *Let A be any subset of Σ^*. The following are equivalent.*

(1) $A \in \mathrm{P}$.
(2) There exists a polynomial p such that for almost every $x \in \Sigma^*$: $\mathrm{ic}^p(x : A) \leqslant \mathrm{C}^A(x)$.

From this characterization, we derive a notion of hard instances.

Definition 8 *Let A be a set, and let q be a polynomial.*

(1) The hard instances w.r.t. q *for A are* $H_q(A) := \{x \mid \mathrm{ic}^q(x : A) > \mathrm{C}^A(x)\}$.
(2) The hard instances *for A are* $H(A) := \bigcap_{k \in \mathbb{N}} H_{n^k+k}(A)$.

Note that $H_q(A) \supseteq H_r(A)$, if $q(n) \leqslant r(n)$ (for every n). This certifies that $H(A)$ is infinite if all $H_q(A)$ are so. Using the definition of hard instances, Theorem 7 says that a set A is not in P iff its set of hard instances $H(A)$ is infinite.

4 Structural properties of hard instances

We want to compare hard instances to complexity cores. The notion of *(polynomial) complexity core* was introduced by Lynch [11] for instances which witness that a set is not polynomial time decidable. In [11] it is shown that every recursive set not in P has an infinite complexity core. Orponen et al. [14] characterized complexity cores in terms of instance complexity.

Theorem 9 [14, Proposition 3.3] *Let A be a recursive set. An infinite set X is a complexity core for A iff for every polynomial p, constant c and almost every $x \in X$, $\mathrm{ic}^p(x : A) > c$.*

By Proposition 1 it follows that every infinite $H(A)$ is a complexity core for A. On the other hand, not every element of a complexity core must be a hard instance, what follows from the "almost every" condition. We want to show that our notion of hard instances is stronger than the notion of complexity cores. In order to do this, we present a set which has a maximal complexity core of much higher density than its set of hard instances.

We say that a set A has *few hard instances*, if $H(A)$ is sparse. There are complexity classes which have sets with few hard instances only. A simple example is the class P: by Theorem 7 every $A \in \mathrm{P}$ has only finitely many hard instances. But also sets not in P may have few hard instances only. For example, the class $\mathrm{IC}[log, poly]$ defined in [14], which consists of sets A having instance complexity $\mathrm{ic}^p(x : A) \leqslant c \log |x|$ for some polynomial p, constant c, and every x. Since for every n, there are at most n^c strings x of length $|x| = n$ having Kolmogorov complexity $\mathrm{C}^A(x) \leqslant c \log |x|$, each set in $\mathrm{IC}[log, poly]$ has at most n^c hard instances of length n.

Proposition 10 *Every set in* $\mathrm{IC}[log, poly]$ *has few hard instances.*

Sparse sets also have few hard instances only. Note that $\mathrm{IC}[log, poly]$ and the class of sparse sets are not comparable.

Proposition 11 *Every sparse set has few hard instances.*

PROOF. Since $\mathrm{SPARSE} \subseteq R^p_c(\mathrm{TALLY})$ [4], there exists a tally set T with $A \leq^p_c T$ via f. Note that f can be chosen to map to sets consisting of strings in 0^* only – since all other strings are not in T – and all those strings have logarithmic Kolmogorov complexity. Let π_y be a program which rejects input x if $y \in f(x)$, and halts undecided otherwise. If $y \notin T$, then π_y is consistent with A. Each instance $x \notin A$ is decided by some π_y for $y \in f(x) - T$. Note that π_y is polynomially time bounded, and $y \in 0^*$. Hence, there exist a polynomial p and a constant c such that $\mathrm{ic}^p(x : A) \leqslant c \log |x|$ for all $x \notin A$. Since the number of strings x of length n with $\mathrm{C}^A(x) \leqslant c \log n$ is at most n^c, there are at most n^c hard instances of length n in $\overline{A}$. Since A is sparse, the number of all hard instances must be sparse too. $\square$

Now we are ready to show that hard instances are a stronger notion of hardness than complexity cores.

Theorem 12 *There exists a set A with complexity core Σ^* and few hard instances.*

PROOF. Balcázar, Díaz, and Gabarró [2, Vol.II, Theorem 6.2] construct a sparse set A which has Σ^* as complexity core. By Proposition 11, A has few hard instances. $\square$

Orponen and Schöning [15] showed that NP-hard sets cannot have sparse complexity cores only, unless P = NP. We consider the same question for hard instances. The main result of this section shows that sets with few hard instances reduce to sparse sets.

Theorem 13 *Let A be a set. If for some polynomial p, the set of hard instances $H_p(A)$ is sparse, then A is in $R^p_{2\text{-}tt}(R^p_c(SPARSE))$.*

PROOF. Let A be a set and p be a polynomial, and assume $|H_p(A) \cap \Sigma^n| \leqslant n^a$ for a constant a and almost every n. The proof idea is as follows. We consider the set of instances which can be consistently decided in polynomial time by programs of logarithmic size. It turns out, that all but a sparse set can. All the instances of logarithmic instance complexity disjunctively reduce to a tally set (essentially an encoding of the small programs), and the sparse rest of the instances many-one reduces to its intersection with A. Putting both reductions together, we get the desired result.

Formally, let $N_{m,k}$ be the following Turing machine, which takes no input, uses oracle A, and computes some output.

$\Pi := \{\pi \in \Sigma^{\leqslant (a+3)\log m} \mid \forall x \in \Sigma^m : \pi(x)_p \simeq A(x)\}$
$T := \{y \in \Sigma^m \mid \text{for all } \pi \in \Pi : \pi(y)_p = \perp\}$
if $|T| \geqslant k$
  then output the k-th element of T w.r.t. lexicographical order
end

$N^A_{m,k}$ halts for all m and k. Let $T(m)$ and $\Pi(m)$ denote the contents of the set variables T and Π when $N^A_{m,k}$ halted. For every m and every $y \in T(m)$ it holds that

$$\mathrm{ic}^p(y : A) > (a+3)\log m$$

by the definition of $\Pi(m)$ and $T(m)$.

The size $|N_{m,k}|$ of $N_{m,k}$ depends mainly on the sizes of m and k. I.e. there exists a constant c, such that for all m, k it holds that $|N_{m,k}| = c + |m| + |k|$. Thus, if y is the output of $N^A_{m,k}$, then $\mathrm{C}^A(y) \leqslant c + |m| + |k|$. Let y be one of the lexicographically first m^{a+1} elements of $T(m)$. Then

$$\mathrm{C}^A(y) \leqslant c + |m| + |m^{a+1}| \leqslant (a+3)\log m$$

for almost every m. Combining both inequalities, we get

$$\mathrm{ic}^p(y : A) > \mathrm{C}^A(y)$$

for almost every m and every y of the lexicographically first m^{a+1} elements of $T(m)$. Thus, each of the lexicographically first m^{a+1} elements of $T(m)$ is in $H_p(A) \cap \Sigma^m$, which contains at most m^a elements. Therefore $|T(m)| \leqslant m^a$ for almost every m.

Let $S = \bigcup_m T(m) \cap A$. Then S is a sparse set. Let U denote an encoding of $\bigcup_m \Pi(m)$ into a tally set, e.g. $U = \{0^{\langle m,q \rangle} \mid m \geqslant 0, q \in \Pi(m)\}$.

Then for every $x \in \Sigma^*$,

$$\begin{aligned} x \in A &\Leftrightarrow \exists \pi \in \Pi(|x|) : \pi(x) \text{ accepts in time } p(|x|) \text{ , or } x \in S \\ &\Leftrightarrow \{0^{\langle |x|, \pi \rangle} \mid \pi \in \Sigma^{\leqslant (a+3)\log |x|}, \pi(x)_p = 1\} \cap U \neq \emptyset \text{ , or } x \in S \quad . \end{aligned}$$

Thus, A can be decided with two queries to different oracles: one query to a set which disjunctively reduces to the tally set U, and one query to the sparse set S. Because the class of sparse sets is closed under marked union, and because $R^p_d(\mathrm{TALLY}) = \text{co-}R^p_c(\mathrm{TALLY})$, this yields that $A \in R^p_{2\text{-}tt}(R^p_c(\mathrm{SPARSE}))$. □

As a consequence, hard sets for NP cannot have few hard instances only, unless P = NP.

Theorem 14 *No $\leq^p_{btt}$-hard set for* NP *has few hard instances, unless* P = NP.

PROOF. Let A be a $\leq^p_{btt}$-hard set for NP, and assume that $H(A)$ is sparse. Then there exist a polynomial p and a sparse set S such that $H_p(A) \subseteq S$. Thus by Theorem 13 it follows that $A \in R^p_{2\text{-}tt}(R^p_c(\mathrm{SPARSE}))$. By a result in [1], no $\leq^p_{btt}$-hard set for NP is in $R^p_{2\text{-}tt}(R^p_c(\mathrm{SPARSE}))$, unless P = NP.

Relaxing from bounded to Turing reductions yields a weaker collapse consequence.

Theorem 15 *No $\leq_T^p$-hard set for* NP *has few hard instances, unless* PH = ZPP$^{\mathrm{NP}}$.

PROOF. By the above argument it follows that NP $\subseteq R_T^p$(SPARSE), if a Turing hard set A for NP has few hard instances. From Köbler and Watanabe [8] it then follows that the polynomial-time hierarchy collapses to ZPP$^{\mathrm{NP}}$.

Acknowledgements

The author would like to thank for the comments of two referees. The author also likes to thank Judy Goldsmith for her hospitality and the research environment provided during his visit at University of Kentucky.

References

[1] V. Arvind, J. Köbler, and M. Mundhenk. On bounded truth-table, conjunctive, and randomized reductions to sparse sets. In *Proceedings 12th Conference on the Foundations of Software Technology & Theoretical Computer Science*, pages 140–151. Lecture Notes in Computer Science #652, Springer-Verlag, 1992.

[2] J.L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I/II.* EATCS Monographs on Theoretical Computer Science. Springer Verlag, 1988/1990.

[3] L. Berman and J. Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM Journal on Computing*, 6(2):305–322, 1977.

[4] H. Buhrman, L. Longpré, and E. Spaan. SPARSE reduces conjunctively to TALLY. In *Proc. 8th Structure in Complexity Theory Conference*, pages 208–214. IEEE, 1993.

[5] H. Buhrman and P. Orponen. Random strings make hard instances. In *Proc. 9th Structure in Complexity Theory Conference*, pages 217–222. IEEE, 1994.

[6] L. Fortnow and M. Kummer. On resource-bounded instance complexity. *Theoretical Computer Science*, 161:123–140, 1996.

[7] J. Hartmanis. Generalized Kolmogorov complexity and the structure of feasible computations. In *Proc. 24th IEEE Symp. on Foundations of Computer Science*, pages 439–445, 1983.

[8] J. Köbler and O. Watanabe. New collapse consequences on NP having small circuits. *SIAM Journal on Computing.* To appear.

[9] M. Kummer. The instance complexity conjecture. In *Proc. 10th Structure in Complexity Theory Conference*, pages 111–124. IEEE, 1995.

[10] M. Li and P. Vitányi. *An introduction to Kolmogorov complexity and its applications.* Springer-Verlag, 1993.

[11] N. Lynch. On reducibility to complex or sparse sets. *Journal of the ACM*, 22:341–345, 1975.

[12] S. Mahaney. Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis. *Journal of Computer and System Sciences*, 25(2):130–143, 1982.

[13] M. Mundhenk. NP-hard sets have may hard instances. In *Proc. 25th Mathematical Foundations of Computer Sciences*, pages 428–437. Lecture Notes in Computer Science #1295, Springer-Verlag, 1997.

[14] P. Orponen, K. Ko, U. Schöning, and O. Watanabe. Instance complexity. *Journal of the ACM*, 41:96–121, 1994.

[15] P. Orponen and U. Schöning. The structure of polynomial complexity cores. In *11th Symp. on Mathematical Foundations of Computer Science.* Lecture Notes in Computer Science #176, Springer-Verlag, 1984.

[16] U. Schöning. *Complexity and structure*, volume 211 of *Lecture Notes in Computer Science.* Springer-Verlag, 1985.