Electronic Colloquium on Computational Complexity, Report No. 58 (2021)

Average-Case Hardness of NP from Exponential Worst-Case Hardness Assumptions

Shuichi Hirahara
National Institute of Informatics
s_hirahara@nii.ac.jp

April 21, 2021

Abstract

A long-standing and central open question in the theory of average-case complexity is to base average-case hardness of NP on worst-case hardness of NP. A frontier question along this line is to prove that PH is hard on average if UP requires (sub-)exponential worst-case complexity. The difficulty of resolving this question has been discussed from various perspectives based on technical barrier results, such as the limits of black-box reductions and the non-existence of worst-case hardness amplification procedures in PH.

In this paper, we overcome these barriers and resolve the open question by presenting the following main results:

1. $\mathsf{UP} \not\subseteq \mathsf{DTIME}\left(2^{O(n/\log n)}\right)$ implies $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$.
2. $\mathsf{PH} \not\subseteq \mathsf{DTIME}\left(2^{O(n/\log n)}\right)$ implies $\mathsf{DistPH} \not\subseteq \mathsf{AvgP}$.
3. $\mathsf{NP} \not\subseteq \mathsf{DTIME}\left(2^{O(n/\log n)}\right)$ implies $\mathsf{DistNP} \not\subseteq \mathsf{Avg_P P}$. Here, $\mathsf{Avg_P P}$ denotes P-*computable average-case polynomial time*, which interpolates average-case polynomial-time and worst-case polynomial-time. We complement this result by showing that $\mathsf{DistPH} \not\subseteq \mathsf{AvgP}$ if and only if $\mathsf{DistPH} \not\subseteq \mathsf{Avg_P P}$.

At the core of all of our results is a new notion of *universal heuristic scheme*, whose running time is P-computable average-case polynomial time under every polynomial-time samplable distribution. Our proofs are based on the meta-complexity of time-bounded Kolmogorov complexity: We analyze average-case complexity through the lens of worst-case meta-complexity using a new "algorithmic" proof of language compression and weak symmetry of information for time-bounded Kolmogorov complexity.

Contents

ISSN 1433-8092

1 Introduction

Understanding the average-case complexity of NP is fundamental in the theory of computation. The average-case complexity reflects the performance of an algorithm in practice better than the worst-case complexity does. The theory of NP-completeness [Coo71, Kar72, Lev73] has identified many natural problems as NP-complete problems, which are considered to be "intractable" problems. However, NP-completeness is defined in terms of *worst-case* complexity. The difficulty of NP-complete problems is measured on contrived instances that are produced from reductions of NP-completeness proofs; such instances may differ significantly from real-world instances. Exploiting this disparity, researchers have developed efficient algorithms for several NP-complete problems that run in *expected polynomial time* with respect to natural distributions on instances. For example, the Hamiltonian path problem can be solved in expected linear time with respect to the Erdős-Rényi random graph (see, e.g., [GS87, Tho89, AK20]). Another motivation to study the average-case complexity of NP stems from cryptography: The security of complexity-theory-based cryptographic primitives is based on the average-case hardness of NP [IL89]. The question of whether there is a public-key cryptosystem whose security is based on the worst-case hardness of NP-complete problems dates back to as early as 1970s when Diffie and Hellman [DH76] introduced the notion of public-key cryptography. These studies motivate the following question:

Question. Can the average-case hardness of NP be based on the worst-case hardness of NP?

Average-case complexity theory, whose foundation was laid by Levin [Lev86] and other researchers [BCGL92, IL90, Imp95, BT06a], enables formalizing the question. Following the notations standardized by Bogdanov and Trevisan [BT06a], we briefly review the basic concepts.

1.1 Average-Case Complexity

In average-case complexity theory, we are concerned with a *distributional problem*, which is a pair $(L, \mathcal{D})$ of a decision problem[1] $L\colon \{0,1\}^* \to \{0,1\}$ and a family $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ of distributions on instances. Here, the subscript n means the "instance size" (which may not be equal to the length of an instance). We say that a distributional problem $(L, \mathcal{D})$ is solvable in *average-case polynomial time* if there exists an algorithm A such that A solves L on every instance in the support of $\mathcal{D}$, and there exists a constant $\epsilon > 0$ such that, for every $n \in \mathbb{N}$, it holds that $\mathbb{E}_{x\sim\mathcal{D}_n}[t(x)^\epsilon] \le n^{O(1)}$, where $t\colon \{0,1\}^* \to \mathbb{N}$ is an upper bound of the running time of A on input x.[2] We denote by AvgP the class of distributional problems that admit average-case polynomial-time algorithms.

In practice, it is highly desirable that an upper bound $t(x)$ of the running time is efficiently computable for every instance x. Knowing the upper bound $t(x)$ of the running time of a heuristic algorithm beforehand enables one to save computational resources by avoiding the computation of a solution for an instance x if $t(x)$ is too large. A trivial way to estimate the running time of a heuristic algorithm is to simply run the algorithm and wait until it halts; however, it is tedious to wait for a heuristic algorithm that may take an exponential time to halt. This motivates us to introduce a class $\mathsf{Avg_P P}$ $(\subseteq \mathsf{AvgP})$ of heuristic algorithms whose running time can be efficiently estimated. Specifically, a distributional problem is said to be solvable in P-*computable average-case polynomial*

[1] We identify a language $L \subseteq \{0,1\}^*$ with a decision problem $L\colon \{0,1\}^* \to \{0,1\}$.

[2] The presence of ϵ in the definition is highly non-trivial but is required to make the notion of average-case polynomial time robust and broad. There is an equivalent and more intuitive definition of AvgP, which is called an *errorless heuristic scheme* [Imp95, BT06a].

time if, in addition to the above definition of AvgP, the function $t\colon \{0,1\}^* \to \mathbb{N}$ is computable in polynomial time. The class of distributional problems that admit P-computable average-case polynomial-time algorithms is denoted by $\mathsf{Avg_P P}$. (Equivalent definitions of AvgP and $\mathsf{Avg_P P}$ are *errorless heuristic scheme* and P-*bounded failure heuristic scheme*, respectively; see Section 9.) For example, for the HamiltonianPath problem and the Erdős-Rényi random graph $\mathcal{G}(n, 1/2)$, it is not difficult to observe that the distributional problem $(\textsc{HamiltonianPath}, \{\mathcal{G}(n, 1/2)\}_{n\in\mathbb{N}})$ is in $\mathsf{Avg_P P}$ using the heuristic algorithms of [Tho89] (see Appendix B).

An average-case analogue of NP is denoted by DistNP, which consists of distributional problems $(L, \mathcal{D})$ such that $L \in \mathsf{NP}$ and $\mathcal{D} \in \textsc{PSamp}$, where PSamp is the class of polynomial-time samplable distributions. In other words, we focus on analyzing the average-case complexity of NP with respect to the distributions from which a random instance can be efficiently generated. More generally, we define $\mathsf{Dist}(\mathfrak{C})$ to be $\mathfrak{C} \times \textsc{PSamp}$ for any complexity class $\mathfrak{C}$.

A central open question in average-case complexity theory is to connect the average-case hardness of NP to the worst-case hardness of NP:

Open Question 1.1. Does $\mathsf{P} \neq \mathsf{NP}$ imply $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$?

This is arguably one of the four central questions[3] in complexity theory and cryptography. In his influential work, Impagliazzo [Imp95] classified the ultimate consequences of complexity theory into five possible scenarios. Excluding any one of the scenarios is considered an important milestone. Open Question 1.1 and its variants correspond to excluding Heuristica (i.e., a world where NP is hard in the worst case but easy on average) from the five possible worlds.

Currently, a question much weaker than Open Question 1.1 is open. A frontier question along the lines of Open Question 1.1 is to prove the average-case hardness of the polynomial-time hierarchy (PH) assuming the existence of an exponentially hard problem in UP.

Frontier Question 1.2. Does $\mathsf{UP} \not\subseteq \mathsf{DTIME}(2^{o(n)})$ imply $\mathsf{DistPH} \not\subseteq \mathsf{AvgP}$?

Recall that UP $(\subseteq \mathsf{NP})$ is the class of languages L for which there exists a polynomial-time verifier that accepts at most one certificate for every input. The worst-case complexity of UP is known to characterize the existence of a one-to-one one-way function that is hard to invert in the worst case [Ko85, GS88], and there are candidate one-to-one one-way functions conjectured to be exponentially hard to invert even on average (see, e.g., [GS88, GLN11]); thus, the assumption of Frontier Question 1.2 is plausible. (In fact, it is plausible that $\mathsf{UP} \not\subseteq \mathsf{DTIME}(2^{n^2})$ because the current best deterministic upper bound on UP is $\mathsf{EXP} := \bigcup_{c\in\mathbb{N}} \mathsf{DTIME}(2^{n^c})$.)

Because of the inability to resolve Frontier Question 1.2, researchers have pursued formal explanations of the difficulty of this question, and a large body of research (e.g., [FF93, BT06b, AGGM06, BB15, MX10, HMX10, HW20, Vio05b, Vio05a, Imp11, Wat12]) has been devoted to understanding which proof techniques are insufficient for resolving Frontier Question 1.2. We review three lines of research in Section 1.2.

[3]Is $\mathsf{P} \neq \mathsf{NP}$? Does $\mathsf{P} \neq \mathsf{NP}$ imply $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$? Does $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$ imply the existence of a one-way function? Does the existence of a one-way function imply the existence of a public-key cryptosystem? Each question corresponds to excluding one possible world.

1.2 Obstacles to Worst-Case-to-Average-Case Connections

1.2.1 "Non-Existence" of Worst-Case Hardness Amplification Procedures in PH

A general approach for establishing a worst-case-to-average-case connection is to construct a *worst-case hardness amplification procedure.* A (worst-case) hardness amplification procedure $\mathrm{Amp}^{(\text{-})}$ takes a worst-case hard function $f\colon \{0,1\}^n \to \{0,1\}$ and returns an average-case hard function $\mathrm{Amp}^f\colon \{0,1\}^{n^{O(1)}} \to \{0,1\}$. It was shown in a line of study that there exists a PSPACE-computable oracle procedure $\mathrm{Amp}^{(\text{-})}$ that realizes the worst-case hardness amplification procedure (see, e.g., [STV01, TV07] and references therein). Such a procedure enables connecting worst- and average-case complexity for large complexity classes, such as PSPACE and EXP; for example, $\mathsf{Dist}(\mathsf{PSPACE}) \not\subseteq \mathsf{AvgP}$ is known to be equivalent to $\mathsf{PSPACE} \neq \mathsf{P}$ [KS04].

Given the general proof techniques described above, it is natural to hope to resolve Frontier Question 1.2 by constructing a worst-case hardness amplification procedure $\mathrm{Amp}^{(\text{-})}$ such that $\mathrm{Amp}^f \in \mathsf{PH}$ holds for every $f \in \mathsf{UP}$, which can be ensured if $\mathrm{Amp}^{(\text{-})}$ is a PH-computable oracle procedure. However, this hope has been strongly dashed by Viola [Vio05b, Vio05a], who presented two barriers. The first barrier [Vio05b] rules out the existence of a PH-computable hardness amplification procedure $\mathrm{Amp}^{(\text{-})}$ such that the relationship between the worst-case complexity of f and the average-case complexity of Amp^f is proved by a black-box reduction. The second barrier [Vio05a] shows that the existence of a PH-computable hardness amplification procedure $\mathrm{Amp}^{(\text{-})}$ (without any additional assumption) implies the existence of an average-case hard function in PH *unconditionally*, in which case $\mathrm{Amp}^{(\text{-})}$ is ineffective.

Theorem 1.3 (Viola [Vio05a]; informal)**.** *Assume that there exists a PH-computable oracle machine $\mathrm{Amp}^{(\text{-})}$ such that, for every $f\colon \{0,1\}^n \to \{0,1\}$ that is worst-case hard for circuits of size $2^{0.99n}$, $\mathrm{Amp}^f\colon \{0,1\}^{n^{O(1)}} \to \{0,1\}$ is average-case hard for polynomial-size circuits. Then, there exists a function $f\colon \{0,1\}^{n^{O(1)}} \to \{0,1\}$ in PH that is average-case hard for polynomial-size circuits.*

Another interpretation of Theorem 1.3 is that proving that $\mathsf{UP} \not\subseteq \mathsf{DTIME}(2^{0.99n})$ implies $\mathsf{DistPH} \not\subseteq \mathsf{AvgP}$ via a hardness amplification procedure in PH is at least as hard as resolving the grand challenge of complexity theory, i.e., $\mathsf{P} \neq \mathsf{NP}$.

1.2.2 Limits of Black-Box Reductions

Another natural approach for establishing a worst-case-to-average-case connection such as Open Question 1.1 is to construct (black-box) *reductions.* Specifically, the approach is to construct a reduction that takes an oracle that can solve some distributional problem in DistNP on average and solves NP in the worst case. Usually, the correctness of a reduction can be established for *every oracle* that solves a problem in DistNP regardless of how inefficient the oracle is; such a reduction is referred to as *black-box.* The approach based on black-box reductions has been quite successful for reducing problems in $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$ to DistNP. For example, a line of research (e.g., [Ost91, HILL99, RR97, ABK+06, AD17, AH19, Hir18]) revealed that $\mathsf{SZK} \neq \mathsf{P}$ implies $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$, which can be proved by a black-box reduction.[4] Here, SZK denotes statistical zero knowledge and is a class contained in $\mathsf{AM} \cap \mathsf{coAM} \subseteq \mathsf{NP/poly} \cap \mathsf{coNP/poly}$. A natural hope would be to extend these black-box reduction techniques to NP-complete problems. However, this hope was dashed

[4]Specifically, there is a randomized polynomial-time reduction from every problem in SZK to DistNP. The reduction can be "derandomized" using [BFP05].

by Feigenbaum and Fortnow [FF93] and more strongly by Bogdanov and Trevisan [BT06b]. The latter showed that there exists no black-box randomized polynomial-time nonadaptive reduction from any problem outside $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$ to DistNP. Using a standard padding argument, their results can be generalized to $t(n)$-time reductions.

Theorem 1.4 (Bogdanov and Trevisan [BT06b])**.** *There exists no black-box randomized $t(n)$-time nonadaptive reduction from* UP *to* DistNP *unless* $\mathsf{UP} \subseteq \mathsf{coNTIME}\big(t(n)^{O(1)}\big)/t(n)^{O(1)}$.

This barrier suggests that we cannot hope to use black-box randomized $2^{o(n)}$-time nonadaptive reductions to prove that $\mathsf{UP} \not\subseteq \mathsf{DTIME}\big(2^{o(n)}\big)$ implies $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$.

We note that it is not difficult to prove that $\mathsf{NP} \not\subseteq \mathsf{E} := \mathsf{DTIME}\big(2^{O(n)}\big)$ implies $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$,[5] which can be proved using a $2^{O(n)}$-time black-box reduction. This does not contradict Theorem 1.4 because $\mathsf{NP} \subseteq \mathsf{coNE}/2^{O(n)}$.[6] The challenge is to achieve a time bound below $2^{o(n)}$, in which case it is unlikely that $\mathsf{NP} \subseteq \mathsf{coNTIME}(2^{o(n)})/2^{o(n)}$.

There are two loopholes in the barrier of Theorem 1.4. The first is to use *adaptive* reductions. There are adaptive reductions in the literature (e.g., [HILL99]); however, no known adaptive reduction can cross the barrier of $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$. Moreover, Bogdanov and Brzuska [BB15] extended the barrier result to adaptive reductions in a special case: They showed that there exists no black-box randomized *adaptive* reduction from any problem outside $\mathsf{AM} \cap \mathsf{coAM}$ to the task of inverting size-verifiable one-way functions f on average. Here, we say that f is *size-verifiable* [AGGM06] if the statement $\left|f^{-1}(x)\right| \le \ell$ can be verified in AM given $x \in \{0,1\}^*$ and $\ell \in \mathbb{N}$ as input. The second loophole is to use *non-black-box* reductions. The first non-black-box reduction that crosses the barrier (under a plausible assumption) was given in [Hir18] by exploiting the efficiency of an oracle. This is the approach we take herein.

1.2.3 Relativization Barrier

Yet another type of barriers was presented in [Imp11, Wat12]. A *relativization barrier* is a versatile tool for showing the difficulty of resolving open questions in complexity theory. Most statements in complexity theory can be relativized, which means that a statement remains valid in the presence of an arbitrary oracle. An oracle under which a given statement is not valid indicates the difficulty of proving the statement. Watson [Wat12] constructed an oracle under which there exists no black-box reduction from UP to DistNP, suggesting a need for either non-black-box reductions or non-relativizing proof techniques. Impagliazzo [Imp11] constructed an oracle relative to which there exists no connection between the worst-case hardness of $\mathsf{UP} \cap \mathsf{coUP}$ and the average-case hardness of NP.

Theorem 1.5 (Impagliazzo [Imp11])**.** *There exists an oracle A such that* $\mathsf{DistNP}^A \subseteq \mathsf{AvgP}^A$ *and* $\mathsf{UP}^A \cap \mathsf{coUP}^A \not\subseteq \mathsf{DTIME}\big(2^{n^\alpha}\big)^A$ *for some constant $\alpha > 0$.*

This barrier indicates that a non-relativizing proof technique is required to achieve a time bound below 2^{n^α}.

[5] An even stronger statement that $\mathsf{NE} \neq \mathsf{E}$ implies $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$ holds [BCGL92]. This is because $\mathsf{NE} \neq \mathsf{E}$ is equivalent to the existence of a hard tally language $L \subseteq \{1\}^*$ in NP, which can be solved by making queries to an arbitrary oracle that solves $(L, \mathcal{T}) \in \mathsf{DistNP}$, where $\mathcal{T}$ is the tally distribution.

[6] $\mathsf{coNE}/2^{O(n)}$ contains every language. We note that $\mathsf{NP} \subseteq \mathsf{NE} \subseteq \mathsf{coNE}/O(n)$ also holds [BFS09].

1.3 Our Results

In this work, bypassing all the technical barriers presented above, we prove the following.

Theorem 1.6 (Main)**.**

1. $\mathsf{UP} \not\subseteq \mathsf{DTIME}\big(2^{O(n/\log n)}\big)$ *implies* $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$.
2. $\mathsf{PH} \not\subseteq \mathsf{DTIME}\big(2^{O(n/\log n)}\big)$ *implies* $\mathsf{DistPH} \not\subseteq \mathsf{AvgP}$.
3. $\mathsf{NP} \not\subseteq \mathsf{DTIME}\big(2^{O(n/\log n)}\big)$ *implies* $\mathsf{DistNP} \not\subseteq \mathsf{Avg_P P}$.

Item 3 bases a natural variant of the average-case hardness of NP on the worst-case hardness of NP: DistNP does not admit P-computable average-case polynomial-time algorithms (i.e., heuristic algorithms whose running time can be efficiently estimated) under the worst-case hardness assumption on NP. This is an important step toward Open Question 1.1 in that the notion of P-computable average-case polynomial time naturally interpolates average-case polynomial time and worst-case polynomial time. In terms of Impagliazzo's five worlds, Item 3 excludes a variant of Heuristica, i.e., a world in which $\mathsf{DistNP} \subseteq \mathsf{Avg_P P}$ and $\mathsf{NP} \not\subseteq \mathsf{DTIME}\big(2^{O(n/\log n)}\big)$.

Items 1 and 2 of Theorem 1.6 resolve Frontier Question 1.2. Any proof for these results must circumvent all the technical barriers. It is unlikely that Item 2 can be proved using a worst-case hardness amplification procedure in light of the barrier of Theorem 1.3. Item 1 is quantitatively in the regime to which the barriers of Theorems 1.3 and 1.4 apply (while Theorem 1.5 does not). Therefore, any proof for Item 1 must *overcome* the barriers of Theorems 1.3 and 1.4, while the relativization barrier of Theorem 1.5 can be *bypassed*. A brief explanation of why our proofs are not subject to these barriers is given below:

Non-Black-Box Our worst-case-to-average-case connections are proved by non-black-box reduction techniques that exploit the efficiency of an oracle (i.e., a hypothetical algorithm that solves DistNP). To briefly explain the non-black-box techniques, let $p(n) = n^{O(1)}$ denote the running time of an oracle that solves DistNP problems. The time bound $2^{O(n/\log n)}$ of Theorem 1.6 comes from the fact that p is a "$(1/\epsilon \log n)$-exponential function"[7] for a small constant $\epsilon > 0$, indicating that the $\epsilon \log n$-iterated composition $p^{\epsilon \log n}(n)$ of p is at most $2^{n/\log n}$. This running time suggests that we compose the hypothetical algorithm with itself $\epsilon \log n$ times, thereby exploiting the efficiency of the oracle in an essential manner.

Hardness Amplification Procedure We do not use any standard form of a hardness amplification procedure. However, this is a superficial explanation that our proofs are not subject to the barrier of Theorem 1.3. Viola's proof techniques are applicable beyond the mere statement of Theorem 1.3. In order to truly overcome the barrier, we need to use a proof technique that is not ruled out by *any extension* of Theorem 1.3. Indeed, Viola's proof techniques can be extended to a non-standard form of a hardness amplification procedure on which our proofs are implicitly based.[8]

[7]The name is an analogue of the notion of a half-exponential function.

[8]Specifically, to prove $\mathsf{NP} \not\subseteq \mathsf{DTIME}(2^{o(n)}) \implies \mathsf{DistPH} \not\subseteq \mathsf{AvgP}$, we implicitly consider a procedure $\mathrm{Amp}^f := \mathrm{MINKT}^{\mathsf{NP}^f} \in \mathsf{PH}^f$ for every function f, and show that the existence of an average-case polynomial-time algorithm for $\mathrm{MINKT}^{\mathsf{NP}^f}$ implies $\mathsf{NP}^f \subseteq \mathsf{DTIME}^f\Big(2^{o(n)}\Big)$.

We overcome the barrier by exploiting the fact that the proof technique of Theorem 1.3 makes crucial use of the fact that $\mathrm{Amp}^f\colon \{0,1\}^m \to \{0,1\}$ is a function defined on inputs of a fixed length m. On the other hand, our proofs rely on a procedure $\mathrm{Amp}^f\colon \{0,1\}^* \to \{0,1\}$ that is defined on all inputs.

Relativization Theorem 1.6 is not subject to the relativization barrier of Theorem 1.5 because of the following two reasons.

1. Our proofs do not relativize because we rely on a pseudorandom generator constructed by Buhrman, Fortnow, and Pavan [BFP05] under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$. Their proof uses a version of the PCP theorem, which is known to be a non-relativizing proof technique.
2. The time bound 2^{n^α} given in the relativization barrier is less than our time bound $2^{O(n/\log n)}$.

Investigating which reason is more essential remains an interesting research question.[9]

What is a candidate problem in NP that witnesses $\mathsf{NP} \not\subseteq \mathsf{DTIME}(2^{o(n)})$? Note that n denotes the length of a binary encoding of inputs. The canonical NP-complete problem SAT is not a candidate because an instance of SAT is a binary string of length $n = \Theta(m \log m)$ that encodes an m-clause 3CNF formula on $O(m)$ variables and can be solved in time $2^{O(m)} = 2^{o(n)}$ by an exhaustive search. The *Minimum Circuit Size Problem* (MCSP [KC00]) is the problem of deciding, given a function $f\colon \{0,1\}^n \to \{0,1\}$ encoded as the truth table of length 2^n and a size parameter $s \in \mathbb{N}$, whether f can be computed by a circuit of size s. More generally, for a circuit class $\mathfrak{C}$, $\mathfrak{C}$-MCSP asks to determine whether f can be computed by a $\mathfrak{C}$-circuit of size s. It is known that $\mathfrak{C}$-MCSP is NP-complete for $\mathfrak{C} \in \{\mathsf{DNF}, \mathsf{DNF} \circ \mathsf{XOR}, \mathsf{AC}^0 \text{ formulas}\}$ [Mas79, AHM+08, HOS18, Ila20b]. Using an exhaustive search, $\mathfrak{C}$-MCSP can be solved in time $2^{O(N)}$ on inputs of length $N = 2^n$; however, no algorithm that runs in time $2^{o(N)}$ is known (for any class $\mathfrak{C} \supseteq \mathsf{DNF}$). Ilango [Ila20a] presented an exponential-time reduction from the search version of Formula-MCSP to its decision version, providing some evidence that Formula-MCSP $\notin \mathsf{DTIME}(2^{o(N)})$. Theorem 1.6 bases the $\mathsf{Avg_P P}$-type average-case hardness of NP on such plausible assumptions.

Corollary 1.7. *For every circuit class $\mathfrak{C}$, if $\mathfrak{C}$-MCSP cannot be solved in time $2^{O(N/\log N)}$ on inputs of length N, then $\mathsf{DistNP} \not\subseteq \mathsf{Avg_P P}$ and $\mathsf{DistPH} \not\subseteq \mathsf{AvgP}$.*

Previously, it was shown in [Hir18] that the non-existence of polynomial-time algorithms that solve an approximation version of MCSP implies $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$; however, it is a long-standing open question whether MCSP is NP-complete or not. Corollary 1.7 is the first result that shows some average-case hardness of NP under plausible worst-case hardness assumptions on NP-complete problems, such as DNF-MCSP.[10]

Our actual results are much stronger than the statements of Theorem 1.6. Let $\mathcal{U} = \{\mathcal{U}_n\}_{n\in\mathbb{N}}$ denote the family of the uniform distributions $\mathcal{U}_n$ on $\{0,1\}^n$. Let $\mathcal{T} = \{\mathcal{T}_n\}_{n\in\mathbb{N}}$ denote the *tally*

[9] We mention that Item 2 of Theorem 1.6 can be relativized by replacing the pseudorandom generator construction of [BFP05] with a relativizing proof under the assumption that $\mathsf{DistPH} \subseteq \mathsf{AvgP}$. Whether Items 1 and 3 relativize remains an open question.

[10] The results of [Hir18] do not generalize to $\mathfrak{C}$-MCSP for any class $\mathfrak{C}$ that cannot simulate polynomial-time algorithms because the proof uses a non-black-box reduction that transforms the efficiency of a hypothetical polynomial-time algorithm for DistNP into No instances of MCSP.

distribution, i.e., the family of the distributions $\mathcal{T}_n$ whose support is $\{1^n\}$. We prove that the existence of an *approximately size-verifiable* (e.g., one-to-one or regular) one-way function that is exponentially hard to invert in the worst case implies that NP does not admit any *one-sided-error* heuristic algorithm that correctly computes a $1/\mathsf{poly}(n)$-fraction of inputs without any error on YES instances with respect to either $\mathcal{U}$ or $\mathcal{T}$.

Theorem 1.8. *The following hold for every constant* $\delta > 0$ *and every constant* c.

1'. *If there exists a* $2^{n^{1-\delta}}$*-time approximately size-verifiable*[11] *function that cannot be inverted in time* $2^{O(n/\log n)}$ *in the worst case, then* $\mathsf{coNP} \times \{\mathcal{U}, \mathcal{T}\} \not\subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$.

2'. *If* $\mathsf{PHTIME}\left(2^{n^{1-\delta}}\right) \not\subseteq \mathsf{DTIME}\left(2^{O(n/\log n)}\right)$, *then* $\mathsf{PH} \times \{\mathcal{U}, \mathcal{T}\} \not\subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$.

3'. *If* $\mathsf{AMTIME}\left(2^{n^{1-\delta}}\right) \not\subseteq \mathsf{DTIME}\left(2^{O(n/\log n)}\right)$, *then* $\mathsf{NP} \times \{\mathcal{U}, \mathcal{T}\} \not\subseteq \mathsf{Avg}_{\mathsf{P}}\mathsf{P}$.

Here, $\mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ denotes the class of distributional problems $(L, \mathcal{D})$ for which there exists a polynomial-time algorithm A such that $\Pr_{x\sim\mathcal{D}_n}[A(x;1^n) = L(x)] \geq n^{-c}$ for every $n \in \mathbb{N}$ and $A(x;1^n) = 0$ for every $x \notin L$. Using the characterization of AvgP by an errorless heuristic notion (cf. [BT06b] or Section 9), it is easy to see that $\mathsf{Avg}_{\mathsf{P}}\mathsf{P} \subseteq \mathsf{AvgP} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$. The complexity classes $\mathsf{PHTIME}(t(n))$ and $\mathsf{AMTIME}(t(n))$ denote $t(n)$-time versions of PH and AM, respectively; the assumptions of Theorem 1.8 are quite plausible because the current best deterministic upper bound on these classes is $\mathsf{DTIME}(2^{O(t(n))})$.

It is natural to investigate the situations in which there is a difference between $\mathsf{Avg}_{\mathsf{P}}\mathsf{P}$ and AvgP. In the case of PH, we demonstrate that $\mathsf{DistPH} \subseteq \mathsf{Avg}_{\mathsf{P}}\mathsf{P}$ if and only if $\mathsf{DistPH} \subseteq \mathsf{AvgP}$, suggesting that $\mathsf{Avg}_{\mathsf{P}}\mathsf{P}$ may be somewhat "close" to AvgP.

Theorem 1.9. $\mathsf{DistPH} \not\subseteq \mathsf{Avg}_{\mathsf{P}}\mathsf{P} \iff \mathsf{DistPH} \not\subseteq \mathsf{AvgP} \iff \mathsf{PH} \times \{\mathcal{U}, \mathcal{T}\} \not\subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ *for every constant* c.

Another interpretation of this result is an important step toward establishing the equivalence between $\mathsf{DistPH} \not\subseteq \mathsf{AvgP}$ and $\mathsf{PH} \neq \mathsf{P}$: It is evident that $\mathsf{DistPH} \not\subseteq \mathsf{AvgP} \implies \mathsf{DistPH} \not\subseteq \mathsf{Avg}_{\mathsf{P}}\mathsf{P} \implies \mathsf{PH} \neq \mathsf{P}$. Theorem 1.9 establishes the converse of the first implication.

We also prove that a heuristic algorithm for NP that succeeds on a small fraction of instances can be converted into a heuristic algorithm for UP whose running time can be efficiently estimated.

Theorem 1.10. *If* $\mathsf{coNP} \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ *for some constant* c, *then* $\mathsf{DistUP} \subseteq \mathsf{Avg}_{\mathsf{P}}\mathsf{P}$.

It is often reported that modern SAT solvers work well in practice. Theorem 1.10 suggests that such heuristic algorithms for NP could be used to construct a P-computable average-case polynomial-time algorithm for UP. Taking the contrapositive of Theorem 1.10, we can also regard it as an (average-case-to-average-case) hardness amplification theorem. Bogdanov and Safra [BS07] showed that $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$ implies $\mathsf{DistNP} \not\subseteq \mathsf{Avg}_{1-(\log n)^{-1/10+o(1)}}\mathsf{P}$. Theorem 1.10 provides a much stronger conclusion ($\mathsf{coNP} \times \{\mathcal{U}, \mathcal{T}\} \not\subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$) under an incomparable assumption ($\mathsf{DistUP} \not\subseteq \mathsf{Avg}_{\mathsf{P}}\mathsf{P}$).

[11] We say that a function f is $t(n)$*-time approximately size-verifiable* if f is computable in time $O(t(n))$, where n is the output length of f, and the size of $f^{-1}(f(x))$ can be approximated by an AM protocol on input $f(x)$ within a factor of $t(n)$.

2 Techniques: Meta-Computational Average-Case Complexity

Our proofs are based on the meta-complexity of the time-bounded Kolmogorov complexity. In general, *meta-complexity* refers to the complexity of computing a problem for determining complexity (e.g., the time-bounded Kolmogorov complexity). The approach of analyzing average-case complexity via meta-complexity was recently proposed in [Hir20a] and inspired the present work. Specifically, in [Hir20a], the average-case complexity of PH was exactly characterized by the worst-case meta-complexity of $\mathrm{GapMINKT}^{\mathsf{PH}}$, which is the problem of approximating the PH-oracle time-bounded Kolmogorov complexity (the precise definition is given in Section 2.2). An average-case hardness amplification theorem showing that $\mathsf{DistPH} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P} \iff \mathsf{DistPH} \subseteq \mathsf{AvgP}$ for every constant c was proved as a corollary of the characterization.

Fig. 1 depicts the proof strategy of [Hir20a], which analyzes the average-case complexity of PH through the lens of the worst-case meta-complexity of $\mathrm{GapMINKT}^{\mathsf{PH}}$. In the left-half of the figure are statements on the average-case complexity of PH; in the right-half of the figure are statements on the worst-case meta-complexity of $\mathrm{GapMINKT}^{\mathsf{PH}}$. The average-case hardness amplification theorem is proved by connecting these statements.

$$\begin{tikzcd}
\text{average-case complexity} & \text{worst-case meta-complexity} \\
\mathsf{DistPH} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P} \arrow[r, Rightarrow, "{[\text{Hir18, Hir20b, Hir20a}]}"] & \mathrm{Gap}(\mathrm{K}^{\mathsf{PH}} \text{ vs } \mathrm{K}) \in \mathsf{P} \arrow[d, Rightarrow, "\text{trivial}"] \\
\mathsf{DistPH} \subseteq \mathsf{AvgP} \arrow[u, Rightarrow, "\text{trivial}"'] & \mathrm{GapMINKT}^{\mathsf{PH}} \in \mathsf{P} \arrow[l, Rightarrow, "{[\text{Hir20d, Hir20c, Hir20a}]}"']
\end{tikzcd}$$

Figure 1: The proof strategy of characterizing the average-case complexity of PH by the worst-case meta-complexity of $\mathrm{GapMINKT}^{\mathsf{PH}}$.

We follow an approach similar to [Hir20a]. Fig. 2 depicts an overview of our proof for $\mathsf{DistPH} \subseteq \mathsf{AvgP} \implies \mathsf{PH} \subseteq \mathsf{DTIME}(2^{o(n)})$. In Section 2.1, we present the key notion of universal heuristic

$$\begin{tikzcd}
\mathsf{DistPH} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P} \arrow[r, Rightarrow, "{[\text{Hir18, Hir20b, Hir20a}]}"] \arrow[d, Rightarrow, "\text{Goal}"] & \mathrm{Gap}(\mathrm{K}^{\mathsf{PH}} \text{ vs } \mathrm{K}) \in \mathsf{P} \arrow[d, Rightarrow, "\text{Lemma 2.6}"] \\
\mathsf{PH} \subseteq \mathsf{DTIME}\left(2^{O(n/\log n)}\right) & \begin{array}{c}\forall L \in \mathsf{PH} \text{ admits} \\ \text{universal heuristic schemes}\end{array} \arrow[l, Rightarrow, "\text{Lemma 2.3}"']
\end{tikzcd}$$

Figure 2: A proof strategy for Item 2 of Theorem 1.6.

scheme and prove Lemma 2.3. In Section 2.2, we present the definition of the meta-computational problem $\mathrm{Gap}(\mathrm{K}^{\mathsf{PH}} \text{ vs } \mathrm{K})$ and explain a proof idea of Lemma 2.6.

2.1 Universal Heuristic Scheme

At the core of all of our results is the new concept of a *universal heuristic scheme*. Before presenting the definition of this scheme, let us justify the name. A universal heuristic scheme is shown to be a P-computable average-case polynomial-time algorithm under every $\mathrm{PS}\textsc{amp}$ distribution in the following sense:

Theorem 9.5 (Universality of the Universal Heuristic Scheme)**.** *Assume that* $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ *for some constant c. Then, the following are equivalent for any language L.*

1. *There exists a universal heuristic scheme for L.*
2. $\{L\} \times \mathrm{PS}\textsc{amp} \subseteq \mathsf{Avg}_{\mathsf{P}}\mathsf{P}$.

Toward the definition of a universal heuristic scheme, we review the notion of the time-bounded Kolmogorov complexity. The *t-time-bounded Kolmogorov complexity* of a string $x \in \{0,1\}^*$ is the shortest "size" of a program d that prints x in time t. A formal definition is given by fixing an efficient universal Turing machine U.

Definition 2.1 (Time-bounded Kolmogorov complexity)**.** *For strings $x, y \in \{0,1\}^*$, an oracle $A \subseteq \{0,1\}^*$, and a time bound $t \in \mathbb{N} \cup \{\infty\}$, the A-oracle t-time-bounded Kolmogorov complexity of x given y is defined as*

$$\mathrm{K}^{t,A}(x \mid y) := \min\left\{\, |d| \;\middle|\; U^A \textit{ outputs } x \textit{ on input } (d,y) \textit{ in time } t \,\right\}.$$

We omit the superscript A if $A = \varnothing$, the superscript t if $t = \infty$, and "$\mid y$" if y is the empty string.

The notion of *computational depth* was introduced by Antunes, Fortnow, van Melkebeek, and Vinodchandran [AFvMV06]. The computational depth $\mathrm{cd}^t(x)$ of a string x is defined as $\mathrm{K}^t(x) - \mathrm{K}(x)$ for a time bound t. This is not computable because of the uncomputability of Kolmogorov complexity $\mathrm{K}(\text{-})$. We generalize the notion to *(s,t)-time-bounded computational depth*, which is simply defined as $\mathrm{cd}^{s,t}(x) = \mathrm{K}^s(x) - \mathrm{K}^t(x)$ for a string $x \in \{0,1\}^*$.

A universal heuristic scheme is a heuristic algorithm that runs in time $\mathsf{poly}\left(n, t, 2^{\mathrm{cd}^{t,p(t)}(x)}\right)$ on input x of length n and a parameter t, where p is some polynomial. A formal definition follows.

Definition 6.2 (Universal Heuristic Scheme)**.** *A* universal heuristic scheme *for a language L is a pair (S, C) of polynomial-time algorithms such that, for some polynomial p, for any $n \in \mathbb{N}$, any $t \geq p(n)$, and any $x \in \{0,1\}^n$,*

1. *if $\mathrm{cd}^{t,p(t)}(x) \leq k$, then $C(x, 1^t, 1^k) = 1$, and*
2. *if $C(x, 1^t, 1^k) = 1$, then $S(x, 1^t, 1^{2^k}) = L(x)$.*

S and C are referred to as a solver *and a* checker*, respectively.*

The parameter k determines how long we would like to run the heuristic algorithm. The checker C tests, in time $\mathsf{poly}(|x|, t)$, that the time-bounded computational depth $\mathrm{cd}^{t,p(t)}(x)$ is sufficiently small. If this test is passed, the solver S is guaranteed to compute the correct answer $L(x)$ in time $\mathsf{poly}(|x|, t, 2^k)$.

Now, we state the main lemmas for Theorem 1.6.

Lemma 2.2. *The following hold.*

1. *If* $\mathsf{DistNP} \subseteq \mathsf{AvgP}$*, then every language in* UP *admits a universal heuristic scheme.*
2. *If* $\mathsf{DistPH} \subseteq \mathsf{AvgP}$*, then every language in* PH *admits a universal heuristic scheme.*
3. *If* $\mathsf{DistNP} \subseteq \mathsf{Avg_P P}$*, then every language in* NP *admits a universal heuristic scheme.*

Lemma 2.3. *If there exists a universal heuristic scheme for L, then $L \in \mathsf{DTIME}\big(2^{O(n/\log n)}\big)$.*

Observe that Theorem 1.6 immediately follows from Lemmas 2.2 and 2.3. It is not difficult to verify Lemma 2.3, which we outline below.

Proof Sketch of Lemma 2.3. The idea is to find a parameter t so that an input x of length n is "computationally shallow" in the sense that $\mathrm{cd}^{t,p(t)}(x) \le O(n/\log n)$. Details follow.

Let (S, C) be a universal heuristic scheme for L, and let p be the polynomial of Definition 6.2. Fix any input $x \in \{0,1\}^*$ of length $n \in \mathbb{N}$. The key is the following telescoping sum:

$$\mathrm{K}^{p(n)}(x) - \mathrm{K}^{p^{I+1}(n)}(x) = \mathrm{cd}^{p^1(n),p^2(n)}(x) + \mathrm{cd}^{p^2(n),p^3(n)}(x) + \cdots + \mathrm{cd}^{p^I(n),p^{I+1}(n)}(x),$$

where I is a parameter to be chosen later. The left-hand side is clearly bounded above by $n+O(1)$. (Indeed, $0 \le \mathrm{K}^t(x) \le n + O(1)$ holds for every string x of length n and for all sufficiently large t, as there is an $O(1)$-size program that prints x given x as input.) This means that

$$I \cdot \min\left\{ \mathrm{cd}^{p^i(n),p^{i+1}(n)}(x) \;\middle|\; i \in [I] \right\} \le \mathrm{K}^{p(n)}(x) - \mathrm{K}^{p^{I+1}(n)}(x) \le n + O(1),$$

which implies that there exists $i \in [I]$ such that $\mathrm{cd}^{p^i(n),p^{i+1}(n)}(x) \le (n+O(1))/I =: k$.[12] This naturally motivates the following algorithm: Given an input x of length n, find $i \in [I]$ such that $C(x, 1^{p^i(n)}, 1^k) = 1$, and simulate and output $S(x, 1^{p^i(n)}, 1^{2^k})$. The correctness of the algorithm immediately follows from Item 2 of Definition 6.2. The running time is at most

$$\mathsf{poly}\Big(p^I(n), 2^k\Big) \le 2^{O(n/\log n)},$$

where the last inequality holds by choosing $I := \epsilon \log n$ for a small constant $\epsilon > 0$. □

A formal proof of Lemma 2.3 and its extension are presented in Section 6.

The main innovation of this work is demonstrating the existence of a universal heuristic scheme from heuristic algorithms, i.e., Lemma 2.2. We present two different proof strategies for constructing universal heuristic schemes. One is based on [Hir20a], while the other is based on the work of Antunes and Fortnow [AF09]. The former is used to prove Items 1 and 2 of Lemma 2.2 and is explained in Sections 2.2 and 2.3. The latter is used to prove Item 3 of Lemma 2.2 and is explained in Section 2.4.

[12] $[n]$ denotes $\{1, \ldots, n\}$.

2.2 Universal Heuristic Scheme from Meta-Complexity

We present a proof technique of the following special case of Item 2 of Lemma 2.2.

Theorem 2.4. $\mathsf{DistPH} \subseteq \mathsf{AvgP}$ *implies that every language* $L \in \mathsf{NP}$ *admits a universal heuristic scheme.*

We use the assumption that $\mathsf{DistPH} \subseteq \mathsf{AvgP}$ to obtain an algorithm that solves $\mathrm{GapMINKT}^{\mathsf{NP}}$ in the worst case. Here, $\mathrm{GapMINKT}^{\mathsf{NP}}$ is the problem of approximating the NP-oracle time-bounded Kolmogorov complexity. The complexity of $\mathrm{GapMINKT}^{\mathsf{NP}}$ is referred to as *meta-complexity* because the time-bounded Kolmogorov complexity of a string x itself asks for the complexity of printing a string x. A formal definition of $\mathrm{GapMINKT}^{\mathsf{NP}}$ is as follows:

Definition 2.5 ([Ko91, Hir20b])**.** *For a polynomial* $\tau \colon \mathbb{N} \to \mathbb{N}$ *and an oracle* $A \subseteq \{0,1\}^*$*, define*

$$\Pi^A_{\mathrm{YES}} := \left\{ (x, 1^t, 1^s) \;\middle|\; \mathrm{K}^{t,A}(x) \le s \right\},$$
$$\Pi^A_{\mathrm{NO}} := \left\{ (x, 1^t, 1^s) \;\middle|\; \mathrm{K}^{\tau(|x|+t),A}(x) > s + \log \tau(|x|+t) \right\}.$$

We define $\mathrm{Gap}_\tau\mathrm{MINKT}^A$ *as the promise problem* $(\Pi^A_{\mathrm{YES}}, \Pi^A_{\mathrm{NO}})$*, and* $\mathrm{Gap}_\tau(\mathrm{K}^A \text{ vs } \mathrm{K})$ *as* $(\Pi^A_{\mathrm{YES}}, \Pi^{\varnothing}_{\mathrm{NO}})$*. We say that* $\mathrm{GapMINKT}^A \in \mathsf{P}$ *if there exists some polynomial* τ *such that* $\mathrm{Gap}_\tau\mathrm{MINKT}^A \in \mathsf{P}$*. For a complexity class* $\mathfrak{C}$*, we say that* $\mathrm{GapMINKT}^{\mathfrak{C}} \in \mathsf{P}$ *if* $\mathrm{GapMINKT}^A \in \mathsf{P}$ *for any* $A \in \mathfrak{C}$*. We omit the superscript* A *if* $A = \varnothing$*.*

Using non-black-box reduction techniques, it was shown in [Hir18, Hir20b, Hir20a] that $\mathrm{GapMINKT}^{\mathsf{NP}}$ (and the harder problem $\mathrm{Gap}(\mathrm{K}^{\mathsf{NP}} \text{ vs } \mathrm{K})$) can be solved in the worst case under the assumption that $\mathsf{DistPH} \subseteq \mathsf{AvgP}$.

Lemma 5.1 ([Hir18, Hir20b, Hir20a])**.** *If* $\mathsf{DistPH} \subseteq \mathsf{AvgP}$ *for some constant* c*, then* $\mathrm{Gap}(\mathrm{K}^{\mathsf{NP}} \text{ vs } \mathrm{K}) \in \mathsf{P}$*.*

At the core of Theorem 2.4 is the following lemma.

Lemma 2.6. *If* $\mathrm{Gap}(\mathrm{K}^{\mathsf{NP}} \text{ vs } \mathrm{K}) \in \mathsf{P}$*, then every language* $L \in \mathsf{NP}$ *admits a universal heuristic scheme.*

Lemma 2.6 is based on the DistNP-hardness results of $\mathrm{GapMINKT}^{\mathsf{NP}}$ [Hir20d, Hir20c, Hir20a], which show that $\mathrm{GapMINKT}^{\mathsf{NP}} \in \mathsf{P}$ implies $\mathsf{DistNP} \subseteq \mathsf{AvgP}$. We prove that this AvgP algorithm for DistNP, in fact, realizes a universal heuristic scheme by using a new technical result which we call weak symmetry of information.

Symmetry of information, established by Kolmogorov and Levin [ZL70, Theorem 5.2], is the following fundamental inequality of the Kolmogorov complexity: For every $x \in \{0,1\}^*$ and $w \in \{0,1\}^*$,

$$\mathrm{K}(x, w) \ge \mathrm{K}(x) + \mathrm{K}(w \mid x) - O(\log \mathrm{K}(x, w)).$$

In particular, this implies that, with high probability over a random choice of $w \sim \{0,1\}^m$,

$$\mathrm{K}(x, w) \ge \mathrm{K}(x) + |w| - O(\log \mathrm{K}(x, w)). \qquad (1)$$

We call Eq. (1) *weak symmetry of information.* A time-bounded analogue of symmetry of information was proved by Longpré and Watanabe [LW95] under the assumption that $\mathsf{P} = \mathsf{NP}$. We prove a time-bounded analogue of the weak symmetry of information under the much weaker assumption that NP is easy on average.

Theorem 5.2 (Weak Symmetry of Information)**.** *If* $\mathsf{DistNP} \subseteq \mathsf{AvgP}$*, then there exists a polynomial* p_{w} *such that, for every* $x \in \{0,1\}^*$ *and every* $m \in \mathbb{N}$*, for every* $\epsilon > 0$*, for all sufficiently large* t*,*

$$\Pr_{w \sim \{0,1\}^m}\left[\mathrm{K}^t(xw) \geq \mathrm{K}^{p_{\mathrm{w}}(t/\epsilon)}(x) + m - \log p_{\mathrm{w}}(t/\epsilon)\right] \geq 1 - \epsilon.$$

Our proof of weak symmetry of information is "meta-computational" and fundamentally different from that of [LW95]. On the one hand, the proof of [LW95] is given by translating the original Kolmogorov–Levin proof of symmetry of information to the time-bounded case. On the other hand, our proof relies on a variant of Lemma 5.1, i.e., an efficient algorithm that solves GapMINKT $\in \mathsf{P}$. Our proof is specific to the time-bounded case, and must completely deviate from the Kolmogorov–Levin proof, as there is no algorithm that can compute the resource-unbounded Kolmogorov complexity.

We now sketch the idea of Lemma 2.6. Let $L \in \mathsf{NP}$ and V be a verifier for L; that is, $x \in L$ if and only if $V(x, y) = 1$ for some certificate y. Fix any input x of length n. Let y_x denote the lexicographically first certificate $y_x \in \{0,1\}^{p(n)}$ such that $V(x, y_x) = 1$ (if any), where p is some polynomial. Our proof strategy is to enumerate a list of strings that contains y_x.

A fundamental tool for analyzing the Kolmogorov complexity, identified by [Hir20c], is the notion of *k-wise direct product generator*. A k-wise direct product generator $\mathrm{DP}_k \colon \{0,1\}^{p(n)} \times \{0,1\}^d \to \{0,1\}^{d+k}$ is defined as

$$\mathrm{DP}_k(y_x; z_1, \ldots, z_k) := (z_1, \ldots, z_k, \mathrm{Enc}(y_x)_{z_1}, \ldots, \mathrm{Enc}(y_x)_{z_k}),$$

where Enc is an arbitrary list-decodable error-correcting code, and $\mathrm{Enc}(y_x)_{z_i}$ denotes the z_i-th bit of $\mathrm{Enc}(y_x)$. This is a standard and simple construction of a pseudorandom generator based on the truth table y_x of a hard function. The key insight of [Hir20c] is that DP_k achieves the nearly optimal[13] advice complexity of $k + O(\log n)$, which turned out to be of fundamental importance. To be more specific, we have the following property:

Theorem 3.12 (Reconstruction Property of DP_k [Hir20c]; Informal)**.** *There exists a randomized polynomial-time oracle algorithm* $R^{(\text{-})}$ *(referred to as a reconstruction algorithm) satisfying the following. Let* $D \colon \{0,1\}^{d+k} \to \{0,1\}$ *be an oracle such that* D distinguishes the output distribution $\mathrm{DP}_k(y_x; \text{-})$ from the uniform distribution*; that is,*

$$\left| \Pr_{z \sim \{0,1\}^d}[D(\mathrm{DP}_k(y_x; z)) = 1] - \Pr_{w \sim \{0,1\}^{d+k}}[D(w) = 1] \right| \geq \frac{1}{2}.$$

Then, with high probability over an internal coin flip of R^D*, there exists an advice string* $\alpha \in \{0,1\}^{k+O(\log n)}$ *such that* R^D *outputs* y_x *on input* α*.*

The reconstruction procedure stated above is randomized, whereas we need a deterministic algorithm in the end. However, this is not a problem, as Buhrman, Fortnow, and Pavan [BFP05] showed that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ implies the existence of a nearly optimal pseudorandom generator that

[13] A lower bound on the advice complexity is proved in [TV07].

enables us to derandomize any randomized algorithms (in particular, pr-$\mathsf{BPP} = \mathsf{P}$).[14] Thus, we may assume that the reconstruction procedure R is deterministic, and it is sufficient to design a randomized universal heuristic scheme.

Now, we present the idea behind the universal heuristic scheme for $L \in \mathsf{NP}$. The idea is to construct a distinguisher D using an algorithm for $\mathrm{Gap}(\mathrm{K}^{\mathsf{NP}}$ vs $\mathrm{K})$, and then try all possible advice strings $\alpha \in \{0,1\}^{k+O(\log n)}$ (of Theorem 3.12) to find the certificate y_x. Let M_{K} denote the polynomial-time algorithm that solves $\mathrm{Gap}_\tau(\mathrm{K}^{\mathsf{NP}}$ vs $\mathrm{K})$ for some polynomial τ, and let $p_{\mathrm{K}}(n) := \tau(2n)$. On the one hand, observe that for all sufficiently large t and for every $z \in \{0,1\}^d$,[15]

$$\mathrm{K}^{2t,\mathsf{NP}}(x, \mathrm{DP}_k(y_x; z)) \le \mathrm{K}^t(x) + d + O(\log n) \tag{2}$$

because y_x can be computed in polynomial time given x and oracle access to NP. On the other hand, by the weak symmetry of information, with high probability over a random choice of $w \sim \{0,1\}^{d+k}$,

$$\mathrm{K}^{p_{\mathrm{K}}(2t)}(x, w) > \mathrm{K}^{p_{\mathrm{w}}(p_{\mathrm{K}}(2t))}(x) + d + k - O(\log t). \tag{3}$$

In particular, if $k \ge \mathrm{cd}^{t,p_{\mathrm{w}}(p_{\mathrm{K}}(2t))}(x) + O(\log t)$, the right-hand side of Eq. (2) is smaller than that of Eq. (3). This suggests that we can define D so that

$$D(w) := 1 \quad \Longleftrightarrow \quad M_{\mathrm{K}}\big((x, w), 1^{2t}, 1^s\big) = 1,$$

where $s := \mathrm{K}^t(x) + d + O(\log n)$. With this choice of the parameter, Eq. (2) implies that $(x, \mathrm{DP}_k(y_x; z))$ is a Yes instance of $\mathrm{Gap}_\tau(\mathrm{K}^{\mathsf{NP}}$ vs $\mathrm{K})$ for every z, whereas Eq. (3) implies that (x, w) is a No instance of $\mathrm{Gap}_\tau(\mathrm{K}^{\mathsf{NP}}$ vs $\mathrm{K})$ with high probability over the random choice of w. We conclude that D distinguishes $\mathrm{DP}_k(y_x; \text{-})$ from the uniform distribution.

Now, we can describe a universal heuristic scheme (S, C) for $L \in \mathsf{NP}$. The checker takes $(x, 1^t, 1^k)$ as input and approximately verifies that $\Pr_w[D(w) = 1] \le \frac{1}{4}$ by random sampling. The solver takes $(x, 1^t, 1^{2^k})$ as input, exhaustively searches all the advice strings $\alpha \in \{0,1\}^{k+O(\log n)}$, and accepts if and only if $V(x, y) = 1$ for some string y that is produced by R^D on some advice string α.

The actual proof is more complicated than that presented above because the parameter $s = \mathrm{K}^t(x) + d + O(\log n)$ is not necessarily computable in polynomial time; the exact value of $\mathrm{K}^t(x)$ may be infeasible to compute. This problem can be addressed by approximating the value of $\mathrm{K}^t(x)$ using M_{K}. Extending the proof to all levels of PH makes a formal proof more involved. The formal proof is presented in Section 7.

2.3 Universal Heuristic Scheme for UP

An additional idea is necessary to apply the proof strategy of Section 2.2 to show the average-case hardness of NP. We present a proof idea to show the following:

Theorem 2.7. $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ *implies that every language in* UP *admits a universal heuristic scheme.*

[14] While Lemma 2.6 does not assume $\mathsf{DistNP} \subseteq \mathsf{AvgP}$, a nearly optimal pseudorandom generator can be constructed under the assumption that $\mathrm{Gap}(\mathrm{K}^{\mathsf{NP}}$ vs $\mathrm{K}) \in \mathsf{P}$ [Hir20a].

[15] $\mathrm{K}^{t,\mathsf{NP}}(x)$ is an informal notation that should be interpreted as $\mathrm{K}^{t,\mathsf{SAT}}(x)$.

2.3.1 Algorithmic Language Compression

The *language compression* theorem for the resource-unbounded Kolmogorov complexity refers to the following simple and fundamental fact:

Fact 2.8 (Language Compression). *Let L be a decidable language. Then, for every $n \in \mathbb{N}$ and every $x \in \{0,1\}^n \cap L$,*

$$\mathrm{K}(x) \le \log|L \cap \{0,1\}^n| + O(\log n).$$

The language compression theorem for the *time-bounded* Kolmogorov complexity has been previously studied (e.g., [Sip83, BFL01, BLvM05]). However, few studies have achieved nearly optimal compression. We present a new "algorithmic" proof of language compression that optimally compresses any language in NP under the assumption that NP is easy on average.

Theorem 4.2 (Algorithmic Language Compression). *Let L be a language in NP. Assume that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$. Then, there exists a polynomial p such that a promise problem $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$ defined as*

$$\Pi_{\text{YES}} := L,$$
$$\Pi_{\text{NO}} := \left\{ x \in \{0,1\}^n \;\middle|\; \mathrm{K}^{p(n)}(x) > \log|L \cap \{0,1\}^n| + \log p(n) \right\}$$

is in pr-P.

Why do we call Theorem 4.2 algorithmic language compression? A language compression theorem can be equivalently stated as follows: The promise problem defined in Theorem 4.2 is disjoint (i.e., $\Pi_{\text{YES}} \cap \Pi_{\text{NO}} = \varnothing$). Theorem 4.2 shows not only that Π_{YES} and Π_{NO} can be separated but also that Π_{YES} and Π_{NO} can be separated by an *efficient algorithm* and immediately implies the following:

Corollary 4.4 (Language Compression). *Under the same assumptions of Theorem 4.2, there exists a polynomial p such that $\mathrm{K}^{p(n)}(x) \le \log|L \cap \{0,1\}^n| + \log p(n)$ for every string x of length n.*

Moreover, Theorem 4.2 is a generalization of Lemma 5.1: The worst-case-to-average-case reduction of Lemma 5.1 follows by algorithmically compressing a language $L := \{ (x, 1^t, 1^s) \mid \mathrm{K}^t(x) \le s \}$.

The proof of Theorem 4.2 is based on two lines of research:

1. Buhrman, Lee, and van Melkebeek [BLvM05] identified the relationship between language compression and pseudorandom generators.

2. The k-wise direct pseudorandom generator DP_k was used in [Hir18, Hir20b] in a "meta-computational" way to present a non-black-box worst-case-to-average-case reduction for GapMINKT.

At a very high level, the algorithmic language compression theorem is proved by viewing the idea of [BLvM05] from a meta-computational perspective, as in [Hir18, Hir20b].

2.3.2 Universal Heuristic Scheme Using Algorithmic Language Compression

Using algorithmic language compression, we present an idea for constructing a universal heuristic scheme for every language $L \in \mathsf{UP}$. Let V be a UP-type verifier for L. For parameters k, s, and

t, consider the following language.[16]

$$L' := \left\{ (x, \mathrm{DP}_k(y; z)) \;\middle|\; \mathrm{K}^t(x) \le s \text{ and } \exists y, V(x,y) = 1 \right\}.$$

It is easy to see that $L' \in \mathsf{NP}$. Since V is a UP-type verifier, for every x, the number of certificates y satisfying $V(x,y) = 1$ is at most 1; thus we have $|L'| \le 2^{s+1} \cdot 2^d$, where d is the length of z. Applying Theorem 4.2, we obtain a polynomial-time algorithm M that solves the promise problem

$$\begin{aligned} \Pi_{\mathrm{YES}} &:= L', \\ \Pi_{\mathrm{NO}} &:= \left\{ (x, w) \;\middle|\; \mathrm{K}^{p(t)}(x, w) > s + 1 + d + \log p(t) \right\}. \end{aligned}$$

The remainder of the proof is almost identical to the proof presented in Section 2.2. Given an input $x \in \{0,1\}^*$ and parameters t and $k \in \mathbb{N}$, the goal is to define an oracle D that distinguishes $\mathrm{DP}_k(y_x; \text{-})$ from the uniform distribution, where y_x is the unique certificate for x (if any). We define D so that $D(w) = 1$ if and only if $M(x, w) = 1$. On the one hand, by choosing $s := \mathrm{K}^t(x)$, the definition of L' implies that $(x, \mathrm{DP}_k(y_x; z)) \in L'$ and thus $D(\mathrm{DP}_k(y_x; z)) = 1$ for every z. On the other hand, weak symmetry of information (Theorem 5.2) implies that, with high probability over a random choice of $w \sim \{0,1\}^{d+k}$,

$$\mathrm{K}^{p(t)}(x, w) > \mathrm{K}^{p_{\mathrm{w}}(p(t))}(x) + d + k - O(\log t),$$

which is larger than $s + 1 + d + \log p(n)$ if $k \ge \mathrm{cd}^{t, p_{\mathrm{w}}(p(t))}(x) + O(\log t)$. These two arguments imply that D distinguishes $\mathrm{DP}_k(y_x; \text{-})$ from the uniform distribution, which enables the reconstruction procedure of DP_k to enumerate a list of strings that contains the unique certificate y_x, if any.

We make two remarks on the proof presented above.

1. Why is the proof not applicable to NP or pr-UP? A natural approach is to attempt to use the Valiant–Vazirani theorem [VV86], which shows NP-completeness of pr-UP via a randomized reduction that, given a *fixed* input $x \in \{0,1\}^n$, reduces the number of certificates to 1 with probability $1/\mathsf{poly}(n)$. However, this is not sufficient for our purpose. What we need in the proof above is that the set L' should be small; for example, we need $\#\{ (x,y) \mid V(x,y) = 1, |x| = n \} \le 2^n \cdot \mathsf{poly}(n)$. The notion of approximately size-verifiable functions captures the class of search problems for which the number of solutions can be reduced to $\mathsf{poly}(n)$ using AM-type algorithms *simultaneously for every input* x, which enables extending the aforementioned proof idea to the task of inverting approximating size-verifiable functions.

2. What is the average-case hard problem in DistNP constructed from an exponentially worst-case hard problem in UP? It is implicit in the proof of the algorithmic language compression theorem that the average-case hard problem is to decide whether a given string u sampled from the uniform distribution $\mathcal{U}$ is in the image L'' of L' under the k'-wise direct product generator. Specifically, let $L'' := \{ \mathrm{DP}_{k'}(w; z') \mid w \in L' \}$ for $k' = s + d + O(\log n)$; we show in Theorem 4.2 that L' can be algorithmically compressed if, among other conditions,[17] $(L'', \mathcal{U}) \in \mathsf{AvgP}$ holds.

[16]More formally, L' is defined as an *ensemble* of languages $L'_{\langle n,k,s,t \rangle}$ that are parameterized by n, k, s and t such that $n = |x|$. See Definition 4.1 for the definition of an ensemble of languages.

[17]For example, in Lemma 3.4, we use a tally language in NP as one of the average-case hard problems in DistNP.

2.4 Hardness for P-Computable Average-Case Polynomial Time

The proof idea of Item 3 of Lemma 2.2 is explained below.

Theorem 10.1. *If* $\mathsf{DistNP} \subseteq \mathsf{Avg_P P}$*, then every language in* NP *admits a universal heuristic scheme.*

This corresponds to the direction from Item 2 to Item 1 of Theorem 9.5, which refers to the universality of the universal heuristic scheme; thus, we present the proof idea of Theorem 9.5.

The proof is based on ideas developed by Antunes and Fortnow [AF09], who showed the "universality" of an algorithm that runs in time $2^{\mathrm{cd}^{p(|x|)}(x)+O(\log|x|)}$ on input x, where p is some polynomial. In order to compare their results with ours, we define a variant of the universal heuristic scheme, which we call the AvgP-universal heuristic scheme.

Definition 2.9 (AvgP-Universal Heuristic Scheme; Implicit in [AF09])**.** *An* AvgP-universal heuristic scheme *for a language L is a polynomial-time algorithm S such that, for some polynomial p, for any $n \in \mathbb{N}$, any $t \geq p(n)$, and any $x \in \{0,1\}^n$,*

1. *if* $\mathrm{cd}^t(x) \leq k$*, then* $S(x, 1^t, 1^{2^k}) = L(x)$*, and*
2. $S(x, 1^t, 1^{2^k}) \in \{L(x), \perp\}$.

The notion of AvgP-universal heuristic scheme allows us to encapsulate the theorem of Antunes and Fortnow [AF09] as follows:

Theorem 2.10 (Antunes and Fortnow [AF09])**.** *If* $\mathsf{E} \not\subseteq \mathsf{i.o.DSPACE}(2^{\epsilon n})$ *for some constant* $\epsilon > 0$*, then the following are equivalent for every language L:*

1. *There exists an* AvgP*-universal heuristic scheme for L.*
2. $\{L\} \times \mathrm{PS}\textsc{amp} \subseteq \mathsf{AvgP}$.

The proof of Theorem 2.10 is based on the fundamental relationship between language compression and average-case complexity, which is conceptually simple: Consider an average-case polynomial-time algorithm A with time bound $t\colon \{0,1\}^* \to \mathbb{N}$. Let H be the set of "hard" instances x such that $t(x) \geq 2^k$ for an arbitrary parameter k. Since A is average-case polynomial-time, the size of H is exponentially small in k. Applying the language compression theorem (for the *resource-unbounded* Kolmogorov complexity), any hard instance x in H has low Kolmogorov complexity. This indicates that A can solve any instance x of high Kolmogorov complexity (which, in particular, has small computational depth).

We apply the proof idea of Theorem 2.10 to $\mathsf{Avg_P P}$ algorithms. An essential difference is that we need to use the language compression theorem for the *time-bounded* Kolmogorov complexity (i.e., Corollary 4.4): A universal heuristic scheme must run in time proportional to the exponential of the *time-bounded* computational depth, which indicates that it is inappropriate to use the language compression theorem for the *resource-unbounded* Kolmogorov complexity. Another new ingredient in the proof is Lemma 5.1, which enables estimating the time-bounded Kolmogorov complexity. Details can be found in Section 9.

The hard distribution obtained using the proof idea above is far from the uniform distribution. (The distribution is one that dominates the "time-bounded universal distribution.") In order to make the distribution uniform, we present a different proof that is based on a nearly optimal compression algorithm whose existence we show under the assumption that NP is easy on average. Details can be found in Section 10.

A natural approach to make the distribution uniform is to appeal to the theorem of Impagliazzo and Levin [IL90], who showed that $\mathsf{DistNP} \subseteq \mathsf{AvgZPP}$ if and only if $\mathsf{NP} \times \{\mathcal{U}\} \subseteq \mathsf{AvgZPP}$. However, it is unclear whether their proofs are applicable to $\mathsf{Avg_P P}$. Our proof provides an analogue of [IL90] in the setting of $\mathsf{Avg_P P}$.

2.5 Organization

The remainder of this paper is organized as follows. In Section 3, we present definitions of $\mathsf{Avg}^1\mathsf{P}$, DP_k, and basic properties of Kolmogorov complexity. We present an algorithmic proof of language compression in Section 4 and prove weak symmetry of information in Section 5. We show that the existence of a universal heuristic scheme implies a fast algorithm in Section 6 and construct a universal heuristic scheme for PH and UP in Section 7 and Section 8, respectively. In Section 9, we characterize the notion of P-computable average-case polynomial time using that of P-bounded failure heuristic scheme, and then prove the universality of universal heuristic schemes. In Section 10, we show that the uniform distribution or the tally distribution is the hardest distribution for $\mathsf{Avg_P P}$ to solve DistNP. Section 11 provides the proofs of the main theorems. Finally, Section 12 presents open questions.

3 Preliminaries

Notation Throughout this paper, we often identify $\mathbb{N} \times \mathbb{N}$ with $\mathbb{N}$ using a bijection $\langle \text{-}, \text{-} \rangle \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ defined as, for example, $\langle a, b \rangle := \sum_{i=0}^{a+b} i + a$. Similarly, for any $k \geq 3$, we identify $\mathbb{N}^k$ with $\mathbb{N}$ using a bijection recursively defined as $\langle a_1, a_2, \ldots, a_k \rangle := \langle a_1, \langle a_2, \ldots, a_k \rangle \rangle$. For a distribution $\mathcal{D}$, let $\operatorname{supp}(\mathcal{D})$ denote the support of $\mathcal{D}$; for $x \in \operatorname{supp}(\mathcal{D})$, let $\mathcal{D}(x)$ denote $\Pr_{X \sim \mathcal{D}}[X = x]$. For a family $\mathcal{D} = \{\mathcal{D}_n\}_{n \in \mathbb{N}}$ of distributions, let $\operatorname{supp}(\mathcal{D})$ denote $\bigcup_{n \in \mathbb{N}} \operatorname{supp}(\mathcal{D}_n)$. The bijection is used to regard $\{\mathcal{D}_n\}_{n \in \mathbb{N}}$ as $\left\{\mathcal{D}_{\langle a,b \rangle}\right\}_{a,b \in \mathbb{N}}$.

3.1 Average-Case Complexity Theory

In this subsection, we review the basic concepts of average-case complexity theory and present useful lemmas. Additional details can be found in the survey by Bogdanov and Trevisan [BT06a].

A formal definition of PSamp and $\mathsf{Dist}\mathfrak{C}$ is provided below.

Definition 3.1 (Polynomial-Time Samplable)**.** *We say that a family $\mathcal{D} = \{\mathcal{D}\}_{n \in \mathbb{N}}$ of distributions is* polynomial-time samplable *if there exist a polynomial-time algorithm M and a polynomial p such that, for every $n \in \mathbb{N}$ and every $x \in \{0,1\}^*$,*

$$\Pr_{r \sim \{0,1\}^{p(n)}} [M(1^n, r) = x] = \mathcal{D}_n(x).$$

Let PSamp denote the class of polynomial-time samplable families of distributions.

Definition 3.2. *For a complexity class $\mathfrak{C}$, let $\mathsf{Dist}\mathfrak{C}$ denote the class of distributional problems $(L, \mathcal{D})$ such that $L \in \mathfrak{C}$ and $\mathcal{D} \in \mathrm{PSamp}$.*

One-sided-error heuristics are formally defined as follows.

Definition 3.3 (One-Sided-Error Heuristics)**.** *For a distributional problem $(L, \mathcal{D})$ and a function $\delta\colon \mathbb{N} \to (0,1)$, an algorithm A is said to be a* one-sided-error heuristic algorithm for $(L, \mathcal{D})$ with failure probability δ *if*

1. *$L(x) = 0$ implies $A(x, 1^n) = 0$ for every $n \in \mathbb{N}$ and every $x \in \operatorname{supp}(\mathcal{D}_n)$, and*
2. *$\Pr_{x \sim \mathcal{D}_n}[A(x, 1^n) = L(x)] \geq 1 - \delta(n)$ for every $n \in \mathbb{N}$.*

$\mathsf{Avg}^1_\delta\mathsf{P}$ *denotes the class of distributional problems for which there exists a polynomial-time one-sided-error heuristic scheme with failure probability δ. The* success probability *refers to $1 - \delta$.*

We will use the pseudorandom generator constructed by Buhrman, Fortnow, and Pavan [BFP05] to derandomize randomized algorithms. For a function $G\colon \{0,1\}^s \to \{0,1\}^n$ and $\epsilon > 0$, we say that a circuit D *ϵ-distinguishes* the output distribution of $G(\text{-})$ from the uniform distribution if

$$\left| \Pr_{z \sim \{0,1\}^s}[D(G(z)) = 1] - \Pr_{w \sim \{0,1\}^n}[D(w) = 1] \right| \geq \epsilon.$$

A family $G = \left\{G_n : \{0,1\}^{s(n)} \to \{0,1\}^n\right\}_{n \in \mathbb{N}}$ is said to be a *pseudorandom generator* secure against a class $\mathfrak{C}$ if, for every $C \in \mathfrak{C}$, for all large n, C cannot $1/n$-distinguish $G_n(\text{-})$ from the uniform distribution.

Lemma 3.4 (Buhrman, Fortnow, and Pavan [BFP05]; see also [Hir20a])**.** *If $\mathsf{coNP} \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ for some constant c, then there exists a pseudorandom generator*

$$G = \left\{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\right\}_{n \in \mathbb{N}}$$

computable in time $n^{O(1)}$ and secure against linear-sized circuits.

Proof Sketch. Lemma 3.4 is based on the following four results.

1. $\mathsf{coNP} \times \{\mathcal{T}\} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ implies $\mathsf{NE} = \mathsf{E}$ [BCGL92].
2. $\mathsf{coNP} \times \{\mathcal{U}\} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ implies pr-MA = pr-NP [KS04].
3. If $\mathsf{NE} = \mathsf{E}$ and pr-MA = pr-NP, then $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}(2^{\epsilon n})$ for some constant $\epsilon > 0$ [BFP05] (see also [Hir20a]).
4. If $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}(2^{\epsilon n})$ for some constant $\epsilon > 0$, then there exists a polynomial-time-computable pseudorandom generator secure against linear-sized circuits [IW97].

$\square$

The reason why we consider the tally distribution $\mathcal{T}$ to be one of the hard distributions solely comes from Lemma 3.4. In the remainder of this paper, we primarily consider the uniform distribution. It is convenient to slightly generalize the uniform distribution $\mathcal{U} = \{\mathcal{U}_n\}_{n \in \mathbb{N}}$ as follows.

Definition 3.5 (Parameterized Uniform Distribution)**.** *A family $\mathcal{D} = \{\mathcal{D}_n\}_{n \in \mathbb{N}}$ of distributions is said to be a* parameterized uniform distribution *if there exist efficiently computable functions*[18] *p and $q\colon \mathbb{N} \to \mathbb{N}$ such that $\mathcal{D}_n$ is identical to the distribution that samples $r \sim \{0,1\}^{p(n)}$ and outputs $(r, 1^{q(n)})$.*

[18] We say that a function $p\colon \mathbb{N} \to \mathbb{N}$ is efficiently computable if $p(n)$ is computable on input 1^n in polynomial time.

Lemma 3.6. *Let A be an oracle. Assume that $\mathsf{coNP}^A \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ for some constant c. Then, for every $L \in \mathsf{coNP}^A$ and for every parameterized uniform distribution $\mathcal{D}$, there exists a constant c' such that $(L, \mathcal{D}) \in \mathsf{Avg}^1_{1-n^{-c'}}\mathsf{P}$.*

Proof. The idea is to encode the information of integers p and $q \in \mathbb{N}$ as the length of instances. Specifically, define a language L' so that $x \in L'$ if and only if $\langle p, q\rangle := |x|$ and $(r, 1^q) \in L$, where r denotes the first p bits of x. Observe that $L' \in \mathsf{coNP}^A$. Let M' be a polynomial-time one-sided-error heuristic algorithm that witnesses $(L', \mathcal{U}) \in \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$.

Let p and q be the efficiently computable functions of Definition 3.5. We define a polynomial-time algorithm M so that $M\big((r, 1^{q(n)}), 1^n\big) := 1$ if and only if $M'\big(rr', 1^{\langle p(n), q(n)\rangle}\big) = 1$ for some string $r' := G_m(z)$ in the image of G_m, where $G_m\colon \{0,1\}^{O(\log m)} \to \{0,1\}^{|r'|}$ is the pseudorandom generator of Lemma 3.4 whose output is truncated to $|r'| = \langle p(n), q(n)\rangle - p(n)$ bits, and $m \le \mathsf{poly}(n)$ is a sufficiently large parameter.

Below, we show that M is a one-sided-error heuristic algorithm for $(L, \mathcal{D})$. We first claim that M does not err on No instances. Fix any $n \in \mathbb{N}$. Let $(r, 1^{q(n)}) \in \operatorname{supp}(\mathcal{D}_n) \setminus L$ be a No instance of L. By the definition of L', we have $L'(rr') = 0$ for every $r' \in \{0,1\}^{\langle p(n), q(n)\rangle - p(n)}$; thus, $M'(rr', 1^{\langle p(n), q(n)\rangle}) = 0$, which implies $M\big((r, 1^{q(n)}), 1^n\big) = 0$.

It remains to claim that the success probability of M is at least $n^{-c'}$ for some constant c'. Let ϵ denote the success probability of M'; that is,

$$\Pr_{r,r'}\Big[M'\Big(rr', 1^{\langle p(n), q(n)\rangle}\Big) = L(r)\Big] \ge \epsilon := \langle p(n), q(n)\rangle^{-c},$$

where $r \sim \{0,1\}^{p(n)}$ and $r' \sim \{0,1\}^{\langle p(n), q(n)\rangle - p(n)}$. We analyze the following two cases:

1. Assume that $\Pr_{r,r'}\big[M'\big(rr', 1^{\langle p(n), q(n)\rangle}\big) = 1\big] \ge \epsilon/2$. By an averaging argument, with probability at least $\epsilon/4$ over the choice of r, $\Pr_{r'}\big[M'\big(rr', 1^{\langle p(n), q(n)\rangle}\big) = 1\big] \ge \epsilon/4$, in which case $M\big((r, 1^{q(n)}), 1^n\big) = 1$ follows from the security of the pseudorandom generator G_m. Since M does not err on No instances, it follows that

$$\Pr_{r}\Big[M\Big((r, 1^{q(n)}), 1^n\Big) = L\Big(r, 1^{q(n)}\Big) = 1\Big] \ge \epsilon/4.$$

2. Next, assume that $\Pr_{r,r'}\big[M'\big(rr', 1^{\langle p(n), q(n)\rangle}\big) = 1\big] < \epsilon/2$. In this case, we have

$$\begin{aligned}
&\Pr_{r}\Big[L\Big(r, 1^{\langle q(n)\rangle}\Big) = 0\Big] \\
&\ge \Pr_{r,r'}\Big[M'\Big(rr', 1^{\langle p(n), q(n)\rangle}\Big) = 0\Big] - \Pr_{r,r'}\Big[M'\Big(rr', 1^{\langle p(n), q(n)\rangle}\Big) \ne L\Big(r, 1^{q(n)}\Big)\Big] \\
&\ge 1 - \epsilon/2 - (1 - \epsilon) \ge \epsilon/2.
\end{aligned}$$

Since M does not err on No instances, we obtain

$$\Pr_{r}\Big[M\Big((r, 1^{q(n)}), 1^n\Big) = L\Big(r, 1^{q(n)}\Big) = 0\Big] \ge \epsilon/2.$$

□

3.2 Kolmogorov Complexity and Its Meta-Complexity

We observe two simple facts about Kolmogorov complexity and $\mathrm{Gap}(\mathrm{K}^A \text{ vs } \mathrm{K})$.

Fact 3.7. *For any $s \geq 1$, the number of strings $x \in \{0,1\}^*$ such that $\mathrm{K}(x) < s$ is less than 2^s.*

Proof. The number of programs of length less than s is at most $\sum_{i=0}^{s-1} 2^i < 2^s$. ☐

Fact 3.8. *For every oracle A, the following are equivalent.*

1. $\mathrm{Gap}(\mathrm{K}^A \text{ vs } \mathrm{K}) \in \mathsf{P}$.
2. *There exist a polynomial-time algorithm M and a polynomial p_{K} such that, on input $(x, 1^t)$ with $t \geq |x|$, M outputs an integer v such that*

$$\mathrm{K}^{p_{\mathrm{K}}(t),A}(x) - \log p_{\mathrm{K}}(t) \leq v \leq \mathrm{K}^t(x).$$

Proof. (Item 1 ⇒ 2) Assume that there exists a polynomial-time algorithm M that solves $\mathrm{Gap}_\tau(\mathrm{K}^A \text{ vs } \mathrm{K})$ for some (monotonically increasing) polynomial τ. Define a polynomial-time algorithm M' so that

$$M'(x, 1^t) := \min \left\{ s \in \mathbb{N} \;\middle|\; M(x, 1^t, 1^s) = 1 \right\}.$$

Fix any $x \in \{0,1\}^*$ and $t \leq |x|$, and let v denote the output of $M'(x, 1^t)$. On one hand, since $M(x, 1^t, 1^{\mathrm{K}^t(x)}) = 1$, we have $v \leq \mathrm{K}^t(x)$. On the other hand, since $M(x, 1^t, 1^s) = 0$ holds for every $s \in \mathbb{N}$ such that $\mathrm{K}^{\tau(|x|+t),A}(x) > s + \log \tau(|x|+t)$, we obtain $v \geq \mathrm{K}^{\tau(|x|+t),A}(x) - \log \tau(|x|+t) \geq \mathrm{K}^{p_{\mathrm{K}}(t),A}(x) - \log p_K(x)$, where $p_{\mathrm{K}}(t)$ is defined as $\tau(2t)$.

(Item 2 ⇒ 1) Conversely, given an algorithm M' that satisfies Item 2, we define an algorithm M so that $M(x, 1^t, 1^s) = 1$ if and only if $M'(x, 1^{t'}) \leq s$ for $t' := |x| + t$. If $\mathrm{K}^t(x) \leq s$, then $M'(x, 1^{t'}) \leq \mathrm{K}^{t'}(x) \leq \mathrm{K}^t(x) \leq s$; thus, M accepts. If $\mathrm{K}^{p_{\mathrm{K}}(|x|+t),A}(x) > s + \log p_K(|x|+t)$, then $M'(x, 1^{t'}) \geq \mathrm{K}^{p_{\mathrm{K}}(t'),A}(x) - \log p_K(t') = \mathrm{K}^{p_{\mathrm{K}}(|x|+t),A}(x) - \log p_K(|x|+t) > s$; thus, M rejects. This indicates that M' solves $\mathrm{Gap}_{p_K}(\mathrm{K}^A \text{ vs } \mathrm{K})$. ☐

3.3 k-Wise Direct Product Generator

To simplify later proofs, we construct a k-wise direct generator different from that in [Hir20c]. Specifically, we instantiate a k-wise direct product generator using the Hadamard code.

Definition 3.9 (Hadamard code)**.** *A function* Had *takes a string $x \in \{0,1\}^*$ and maps it to a function $\mathrm{Had}(x)\colon \{0,1\}^{|x|} \to \{0,1\}$ defined as $\mathrm{Had}(x)(y) := \left(\sum_{i=1}^{|x|} x_i y_i\right) \bmod 2$ for every $y \in \{0,1\}^{|x|}$, where x_i denotes the ith bit of x.*

Definition 3.10 (k-Wise Direct Product Generator)**.** *For every $n, k \in \mathbb{N}$, we define the* k-wise direct product generator *to be a function*

$$\mathrm{DP}_k\colon \{0,1\}^n \times \{0,1\}^{nk} \to \{0,1\}^{nk+k}$$

such that

$$\mathrm{DP}_k(x; z^1, \ldots, z^k) := (z^1, \ldots, z^k, \mathrm{Had}(x)(z_1), \ldots, \mathrm{Had}(x)(z_k)).$$

Goldreich and Levin [GL89] showed that the Hadamard code is locally list-decodable.

Lemma 3.11 (Local list-decoding algorithm for the Hadamard code [GL89]; see also [GRS00])**.** *There exists a randomized oracle algorithm M such that M takes n and $\epsilon^{-1} \in \mathbb{N}$ as input and random access to a function $f\colon \{0,1\}^n \to \{0,1\}$, and for every $x \in \{0,1\}^n$ such that*

$$\Pr_{y \sim \{0,1\}^n} [f(y) = \mathrm{Had}(x)(y)] \geq \frac{1}{2} + \epsilon,$$

M^f outputs x with probability at least $1/\mathsf{poly}(n/\epsilon)$ in time $\mathsf{poly}(n/\epsilon)$.

A formal statement of the reconstruction property of DP_k is given below.

Theorem 3.12 (Deterministic Reconstruction for DP_k [Hir20c, Hir20b, Hir20a])**.** *Assume that there exists a pseudorandom generator $G = \left\{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\right\}_{n\in\mathbb{N}}$ computable in time $n^{O(1)}$ and secure against linear-sized circuits. Then, there exist a polynomial-time oracle algorithm $C^{(\cdot)}$ and a polynomial p such that, for every $n \in \mathbb{N}$, $x \in \{0,1\}^n$, parameters $k, \epsilon^{-1}, s \in \mathbb{N}$, and for every randomized circuit D of size s such that*

$$\left|\Pr_{z,r}[D(\mathrm{DP}_k(x;z);r) = 1] - \Pr_{w,r}[D(w;r) = 1]\right| \geq \epsilon,$$

where $z \sim \{0,1\}^{nk}$, $w \sim \{0,1\}^{nk+k}$, and $r \sim \{0,1\}^s$, the algorithm C^D takes $(x, 1^k, 1^{\epsilon^{-1}}, 1^s)$ as input and outputs a program of size at most $k + \log p(ns/\epsilon)$ that prints x given D in time $p(ns/\epsilon)$. In particular,

$$\mathrm{K}^{p(ns/\epsilon)}(x \mid D) \leq k + \log p(ns/\epsilon).$$

Since all strings with low time-bounded Kolmogorov complexity can be enumerated by an exhaustive search, the conclusion of Theorem 3.12 means that, in time $\mathsf{poly}(ns2^k/\epsilon)$, given a description of the circuit D, one can enumerate a list of strings that contains x.

Remark 3.13. The advantage of the k-wise direct product generator DP_k instantiated with the Hadamard code is that DP_k does not depend on the advantage ϵ, where ϵ is the parameter in Theorem 3.12. This will simply our proofs slightly. The construction given in [Hir20c, Hir20b] depends on ϵ because of the choice of an error-correcting code. The disadvantage of DP_k instantiated with Had is that it requires many random bits, which is inappropriate for the purpose of [Hir20c]. This is not a problem for our purpose because of Lemma 3.4.

For completeness, we include a proof of Theorem 3.12. The proof is based on the following.[19]

Lemma 3.14. *For any parameters $n, k \in \mathbb{N}$, and $\epsilon > 0$ with $k \leq 2n$, there exists a pair of algorithms A and $R^{(\cdot)}$ (associated with DP_k) satisfying the following.*

- *R^D takes oracle access to a function $D\colon \{0,1\}^{d+k} \to \{0,1\}$, where $d := nk$.*
- *$A\colon \{0,1\}^n \times \{0,1\}^r \to \{0,1\}^k$ is called an* advice function *and is computable in time $\mathsf{poly}(n/\epsilon)$.*
- *$R^D\colon \{0,1\}^k \times \{0,1\}^r \to \{0,1\}^n$ is called a* reconstruction procedure *and is computable in time $\mathsf{poly}(n/\epsilon)$.*
- *The randomness complexity r is at most $\mathsf{poly}(n/\epsilon)$.*

[19]The formalization of Lemma 3.14 is inspired by [TUZ07].

- *For any string $x \in \{0,1\}^n$ and any function D that ϵ-distinguishes the output distribution of* $\mathrm{DP}_k(x, \text{-})$ *from the uniform distribution, it holds that*

$$\Pr_{w \sim \{0,1\}^r}\left[R^D(A(x,w), w) = x\right] \geq 1/\mathsf{poly}(n/\epsilon).$$

Proof. We use a standard hybrid argument (as in [NW94, Vad12]). Fix any string $x \in \{0,1\}^n$. Let $\widehat{x}$ denote $\mathrm{Had}(x)\colon \{0,1\}^n \to \{0,1\}$. Assume that $D\colon \{0,1\}^{d+k} \to \{0,1\}$ satisfies

$$\Pr_{\bar{z}}\left[D(z^1, \ldots, z^k, \widehat{x}(z^1), \ldots, \widehat{x}(z^k)) = 1\right] - \Pr_{\bar{z}, b}\left[D(z^1, \ldots, z^k, b_1, \ldots, b_k) = 1\right] \geq \epsilon,$$

where $\bar{z} = (z^1, \ldots, z^k) \sim (\{0,1\}^n)^k$ and $b \sim \{0,1\}^k$. For every $i \in \{0, \ldots, k\}$, define the i-th hybrid distribution H_i as the distribution of

$$(z^1, \ldots, z^k, \widehat{x}(z^1), \ldots, \widehat{x}(z^i), b_{i+1}, \ldots, b_k),$$

where $\bar{z} = (z^1, \ldots, z^k) \sim (\{0,1\}^n)^k$ and $b_{i+1}, \ldots, b_k \sim \{0,1\}$. By this definition, H_0 is identically distributed with the uniform distribution, and H_k is a distribution identical to $\mathrm{DP}_k(x, \bar{z})$. Therefore,

$$\mathop{\mathbb{E}}_{\substack{i \sim [k] \\ \bar{z}, b}}\left[D(H_i) - D(H_{i-1})\right] \geq \frac{\epsilon}{k}.$$

By an averaging argument, we obtain

$$\Pr_{\substack{i \sim [k], b \\ z^1, \ldots, z^{i-1}, z^{i+1}, \ldots, z^k}}\left[\mathop{\mathbb{E}}_{z^i \sim \{0,1\}^n}\left[D(H_i) - D(H_{i-1})\right] \geq \frac{\epsilon}{2k}\right] \geq \frac{\epsilon}{2k}. \tag{4}$$

Define the advice function A as $A(x,w) := (\widehat{x}(z^1), \ldots, \widehat{x}(z^{i-1}), b_i, \ldots, b_k) \in \{0,1\}^k$, where w is a coin flip sequence that contains the random choice of $i \sim [k]$, $z^1, \ldots, z^{i-1} \sim \{0,1\}^n$, and $b \sim \{0,1\}^k$. By a standard calculation (see, e.g., [Vad12, Proposition 7.16]), it follows from Eq. (4) that

$$\Pr_{z^i \sim \{0,1\}^n}\left[D(\bar{z}, A(x,w)) \oplus 1 \oplus b_i = \widehat{x}(z^i)\right] \geq \frac{1}{2} + \frac{\epsilon}{2k} \tag{5}$$

with probability at least $\epsilon/2k$ over the random choice of $(i, z^{[k]\setminus\{i\}}, b)$.

Now we describe the reconstruction procedure $R^D(\alpha, w)$. Given an advice string $\alpha \in \{0,1\}^k$ and a coin flip w, we regard the random bits w as $i \sim [k]$, $z^1, \ldots, z^{i-1}, z^{i+1}, \ldots, z^k \sim \{0,1\}^n$, $b \sim \{0,1\}^k$, and $r_0 \sim \{0,1\}^{\mathsf{poly}(n/\epsilon)}$. Define a function $f\colon \{0,1\}^n \to \{0,1\}$ as

$$f(z^i) := D(\bar{z}, \alpha) \oplus 1 \oplus b_i$$

for every $z^i \in \{0,1\}^n$. The reconstruction procedure runs the list-decoding algorithm of Lemma 3.11 using r_0 as a coin flip sequence, and outputs the output of the list-decoding algorithm.

We claim that $\Pr_w\left[R^D(A(x,w), w) = x\right] \geq \epsilon/2kL$ for some $L = \mathsf{poly}(n/\epsilon)$. By Eq. (5), with probability at least $\epsilon/2k$ over the random choice of $(i, z^{[k]\setminus\{i\}}, b)$, $f(z^i)$ and $\widehat{x}(z^i)$ agree on at least a $(1/2 + \epsilon/2k)$-fraction of all the strings $z^i \in \{0,1\}^n$, in which case the list-decoding algorithm outputs x with probability at least $1/\mathsf{poly}(n/\epsilon)$ over the random choice of r_0. Therefore, we obtain

$$\Pr_w\left[R^D(A(x,w), w) = x\right] \geq \frac{\epsilon}{2k} \cdot \frac{1}{\mathsf{poly}(n/\epsilon)}.$$

□

Proof of Theorem 3.12. We may assume without loss of generality that $k \le 2n$. Let D be a randomized circuit D of size s such that

$$\left| \Pr_{z,r}[D(\mathrm{DP}_k(x;z);r)=1] - \Pr_{w,r}[D(w;r)=1] \right| \ge \epsilon.$$

By the security property of the pseudorandom generator G_s, we have the following two inequalities:

$$\left| \Pr_{z,\sigma}[D(\mathrm{DP}_k(x;z);G_s(\sigma))=1] - \Pr_{z,r}[D(\mathrm{DP}_k(x;z);r)=1] \right| \le \frac{\epsilon}{4},$$

$$\left| \Pr_{w,\sigma}[D(w;G_s(\sigma))=1] - \Pr_{w,r}[D(w;r)=1] \right| \le \frac{\epsilon}{4}.$$

It follows from the three inequalities above that

$$\left| \Pr_{z,\sigma}[D(\mathrm{DP}_k(x;z);G_s(\sigma))=1] - \Pr_{w,\sigma}[D(w;G_s(\sigma))=1] \right| \ge \frac{\epsilon}{2}.$$

We define a function $D_\sigma \colon \{0,1\}^{nk+k} \to \{0,1\}$ so that $D_\sigma(w) := D(w;G_s(\sigma))$. Then, there exists a seed $\sigma \in \{0,1\}^{O(\log s)}$ such that

$$\left| \Pr_{z}[D(\mathrm{DP}_k(x;z);G_s(\sigma))=1] - \Pr_{w}[D(w;G_s(\sigma))=1] \right| \ge \frac{\epsilon}{2}.$$

Applying Lemma 3.14 to D_σ, there exists a polynomial p such that

$$\Pr_{w\sim\{0,1\}^r}\left[R^{D_\sigma}(A(x,w),w)=x\right] \ge \frac{1}{p(n/\epsilon)},$$

where $r \le p(n/\epsilon)$. Observe that the condition that $R^{D_\sigma}(A(x,w),w)=x$ can be checked by a circuit of size s' given w as input, where $2p(n/\epsilon) \le s' \le q(ns/\epsilon)$ for some polynomial q. We thus obtain

$$\Pr_{\sigma'}\left[R^{D_\sigma}(A(x,G_{s'}(\sigma')),G_{s'}(\sigma'))=x\right] \ge \frac{1}{2p(n/\epsilon)}.$$

In particular, there exists a seed $\sigma' \in \{0,1\}^{O(\log s')}$ such that $R^{D_\sigma}(A(x,G_{s'}(\sigma')),G_{s'}(\sigma'))=x$.

We are now ready to describe a program M that prints x efficiently given the description of the randomized circuit D: Given an advice string $\alpha := A(x,G_{s'}(\sigma')) \in \{0,1\}^k$ and seeds $\sigma \in \{0,1\}^{O(\log s)}$ and $\sigma' \in \{0,1\}^{O(\log s')}$, the program M computes and outputs $R^{D_\sigma}(\alpha,G_{s'}(\sigma'))$ in time $\mathsf{poly}(ns/\epsilon)$; we thus obtain

$$\mathrm{K}^{\mathsf{poly}(ns/\epsilon)}(x \mid D) \le k + O(\log s) + O(\log s').$$

The algorithm C^D that prints M can be naturally defined as follows: Given $(x,1^k,1^{\epsilon^{-1}},1^s)$ as input, the algorithm finds seeds $\sigma \in \{0,1\}^{O(\log s)}$ and $\sigma' \in \{0,1\}^{O(\log s')}$ such that

$$R^{D_\sigma}(A(x,G_{s'}(\sigma')),G_{s'}(\sigma'))=x$$

by an exhaustive search and outputs M. □

4 Algorithmic Language Compression

In this section, we present an algorithmic proof of the language compression theorem. To state the language compression theorem formally, it is useful to regard a language L as a family $\{L_t\}_{t\in\mathbb{N}}$ of languages that are indexed by a unary 1^t. We call such a family an *ensemble of languages*:

Definition 4.1 (Ensemble of Languages)**.** *For every language $L \subseteq \{0,1\}^*$ and every $t \in \mathbb{N}$, let L_t denote $\{\, x \in \{0,1\}^* \mid (x, 1^t) \in L \,\}$. We say that a language $L \subseteq \{0,1\}^*$ is an* ensemble of languages *if there exists a polynomial p_L such that $|x| \le p_L(t)$ for any $t \in \mathbb{N}$ and any $x \in L_t$. We identify an ensemble $L \subseteq \{0,1\}^*$ of languages with a family $\{L_t\}_{t\in\mathbb{N}}$.*

The following is the main result of this section.

Theorem 4.2 (Algorithmic Language Compression)**.** *Let A be an oracle and $L = \{L_t\}_{t\in\mathbb{N}} \in \mathsf{NP}^A$ be an ensemble of languages. Assume that $\mathsf{coNP}^A \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ for some constant c. Then, there exists a polynomial p such that a promise problem $\Pi = (\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ defined as*

$$\Pi_{\mathrm{YES}} := \left\{\, (x, 1^t) \;\middle|\; x \in L_t \,\right\} = L,$$
$$\Pi_{\mathrm{NO}} := \left\{\, (x, 1^t) \;\middle|\; \mathrm{K}^{p(t)}(x) > \log |L_t| + \log p(t) \,\right\}$$

is in pr-P.

Remark 4.3. If $|L_t| = 0$, we have $(\Pi_{\mathrm{NO}})_t = \{0,1\}^*$ because $\mathrm{K}^{p(t)}(x) > \log |L_t| + \log p(t) = -\infty$ holds for any x.

We call Theorem 4.2 an "algorithmic" proof of language compression because of the following.

Corollary 4.4 (Language Compression)**.** *Under the same assumptions of Theorem 4.2, there exists a polynomial p such that $\mathrm{K}^{p(t)}(x) \le \log |L_t| + \log p(t)$ for every $t \in \mathbb{N}$ and every $x \in L_t$.*

Proof. For any $x \in L_t$, we have $(x, 1^t) \in L = \Pi_{\mathrm{YES}}$. Since $\Pi \in$ pr-P, we must have $\Pi_{\mathrm{YES}} \cap \Pi_{\mathrm{NO}} = \varnothing$; we thus obtain $(x, 1^t) \notin \Pi_{\mathrm{NO}}$, from which the result follows. $\square$

The remainder of this section is devoted to proving Theorem 4.2. We start with a lemma that enables us to estimate $\log |L_t|$.

Lemma 4.5. *Under the same assumptions of Theorem 4.2, there exists a polynomial-time algorithm that, on input 1^t, outputs a value $v \in \mathbb{N}$ such that $|L_t| \le v \le 4|L_t|$.*

The proof idea of Lemma 4.5 is that any problem in pr-AM can be solved on unary inputs under the assumption that NP is easy on average. The lower bound protocol of Goldwasser and Sipser [GS86] enables us to estimate the size of L_t in pr-AM.

Lemma 4.6 (Lower Bound Protocol; Goldwasser and Sipser [GS86]; see also [BT06b, Lemma 2.6])**.** *Let $L \in \mathsf{NP}^A$ and let p be a polynomial. Define $s(x) := \left|\left\{\, y \in \{0,1\}^{p(|x|)} \;\middle|\; (y, x) \in L \,\right\}\right|$ for each $x \in \{0,1\}^*$. Then, a promise problem $\Pi = (\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ defined as*

$$\Pi_{\mathrm{YES}} := \{\, (x, \ell) \mid s(x) \ge \ell \,\},$$
$$\Pi_{\mathrm{NO}} := \{\, (x, \ell) \mid s(x) < (1 - \epsilon)\ell \,\}$$

is in pr-AM^A *for any constant $\epsilon > 0$.*

Proof Sketch. A pr-AM^A protocol operates roughly as follows. A verifier picks a pairwise-independent hash h randomly. A prover sends a string y such that $h(y) = 0$ and $(y, x) \in L$ together with a certificate for $(y, x) \in L$.

□

Proof of Lemma 4.5. Let $s(1^t) := |L_t| \le 2^{p_L(t)}$ for every $t \in \mathbb{N}$. Applying Lemma 4.6 to L, there exist an A-oracle polynomial-time algorithm V^A and a polynomial p such that, for every $K > 0$,

1. if $s(1^t) \ge K$, then $\Pr_{r \sim \{0,1\}^{p(t)}}\left[\exists y \in \{0,1\}^{p(t)}, V^A\big((1^t, K), y, r\big) = 1\right] \ge 1 - 2^{-t}$, and
2. if $s(1^t) \le K/2$, then $\Pr_{r \sim \{0,1\}^{p(t)}}\left[\exists y \in \{0,1\}^{p(t)}, V^A\big((1^t, K), y, r\big) = 1\right] < 2^{-t}$.

Define a language L' so that $L' := \left\{ (r, 1^{\langle t,k \rangle}) \;\middle|\; \exists y \in \{0,1\}^{p(t)}, V^A\big((1^t, 2^k), y, r\big) = 1 \right\} \in \mathsf{NP}^A$. Consider a family $\mathcal{D} = \left\{\mathcal{D}_{\langle t,k \rangle}\right\}_{t,k \in \mathbb{N}}$ of distributions, where $\mathcal{D}_{\langle t,k \rangle}$ is a distribution that samples $r \sim \{0,1\}^{p(t)}$ and outputs $(r, 1^{\langle t,k \rangle})$. Observe that this is a parameterized uniform distribution. It follows from the assumption and Lemma 3.6 that $(\mathsf{co}L', \mathcal{D}) \in \mathsf{Avg}^1_{1-n^{-c'}}\mathsf{P}$ for some constant c'. Let $\neg A$ be a polynomial-time one-sided-error heuristic algorithm for $(\mathsf{co}L', \mathcal{D})$ with success probability $\langle t, k \rangle^{-c'}$. Then, for all sufficiently large $t \in \mathbb{N}$ and all $k \le p_L(t)$, we have the following.

1. If $s(1^t) \ge 2^k$, then[20]

$$\Pr_r\left[A(r, 1^{\langle t,k \rangle}) = 1\right] \ge \Pr_r\left[L'(r, 1^{\langle t,k \rangle}) = 1\right] \ge 1 - 2^{-t}.$$

2. If $s(1^t) \le 2^{k-1}$, then

$$\begin{aligned}\Pr_r\left[A(r, 1^{\langle t,k \rangle}) = 1\right] &\le \Pr_r\left[L'(r, 1^{\langle t,k \rangle}) = 1\right] + \Pr_r\left[L'(r, 1^{\langle t,k \rangle}) \ne A(r, 1^{\langle t,k \rangle})\right] \\ &\le 2^{-t} + 1 - \langle t, k \rangle^{-c'} \le 1 - t^{-d},\end{aligned}$$

   where d is some constant independent of t.

The gap between the probabilities $1 - 2^{-t}$ and $1 - t^{-d}$ can be amplified by running A $\mathsf{poly}(t)$ times; thus, there exists a randomized polynomial-time algorithm that solves the following promise problem $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$:

$$\begin{aligned}\Pi_{\text{YES}} &:= \left\{ 1^{\langle t,k \rangle} \;\middle|\; s(1^t) \ge 2^k \right\}, \\ \Pi_{\text{NO}} &:= \left\{ 1^{\langle t,k \rangle} \;\middle|\; s(1^t) \le 2^{k-1} \right\}.\end{aligned}$$

Since $\Pi \in$ pr-BPP = pr-P by Lemma 3.4, there exists a deterministic polynomial-time algorithm A' that solves Π.

Now we are ready to present a polynomial-time algorithm B that computes an approximation value v of $|L_t|$. Define B so that $B(1^t) := \max\left(\{0\} \cup \left\{ 2^{k+1} \;\middle|\; A'(1^{\langle t,k \rangle}) = 1 \right\}\right)$ for every $t \in \mathbb{N}$.

[20]More precisely, A takes $((r, 1^{\langle t,k \rangle}), 1^{\langle t,k \rangle})$ as input. Since the last input is clearly redundant, we omit it for simplicity.

We claim the correctness of the algorithm B. Fix any $t \in \mathbb{N}$, and let $v := B(1^t)$ and $s := s(1^t)$. If $s = 0$, it is easy to observe that $v = 0$, in which case we have $s \le v \le 4s$, as desired. Consider the case where $s > 0$. Let $k \in \mathbb{N}$ denote $\max \{ k \mid 2^k \le 2s \}$. Since $s \le 2^k$, we have $1^{\langle t, k+1 \rangle} \in \Pi_{\text{No}}$; thus, $v \le 2^{k+1} \le 4s$. Since $2^k \le 2s$, we also obtain $1^{\langle t, k-1 \rangle} \in \Pi_{\text{Yes}}$; thus, $v \ge 2^k \ge s$. We conclude that $s \le v \le 4s$ for every $s \in \mathbb{N}$. □

We are ready to present an algorithmic proof of language compression.

Proof of Theorem 4.2. Let A be the algorithm of Lemma 4.5. Let $k(t) := \log A(1^t) + d \log t$ for some constant d chosen later. Let $L' = \{ (\mathrm{DP}_k(x; z), 1^{\langle n,t \rangle}) \mid x \in L_t \cap \{0,1\}^n, k = k(t) \}$. Since $L \in \mathsf{NP}^A$, one can observe that $L' \in \mathsf{NP}^A$. (Indeed, $(w, 1^{\langle n,t \rangle}) \in L'$ if and only if there exist $x \in \{0,1\}^n$, a certificate y for $x \in L_t$, and $z \in \{0,1\}^{n \cdot k(t)}$ such that $w = \mathrm{DP}_{k(t)}(x; z)$.) Consider a parameterized uniform distribution $\mathcal{D} = \{ \mathcal{D}_{\langle n,t \rangle} \}_{n,t \in \mathbb{N}}$ such that $\mathcal{D}_{\langle n,t \rangle}$ picks $w \sim \{0,1\}^{nk(t)+k(t)}$ and outputs $(w, 1^{\langle n,t \rangle})$. It follows from Lemma 3.6 that $(\mathsf{co}L', \mathcal{D}) \in \mathsf{Avg}^1_{1-n^{-c'}}\mathsf{P}$ for some constant c'. Let $\neg B$ be a polynomial-time one-sided-error heuristic algorithm for $(\mathsf{co}L', \mathcal{D})$ with success probability $\langle n,t \rangle^{-c'}$.

We first claim that the number of Yes instances in L' is relatively small. For each $n, t \in \mathbb{N}$, by a union bound, we have

$$\begin{aligned}
&\Pr_{w \sim \{0,1\}^{nk(t)+k(t)}} \left[(w, 1^{\langle n,t \rangle}) \in L' \right] \\
&= \Pr_{w \sim \{0,1\}^{nk(t)+k(t)}} \left[\exists x \in L_t \cap \{0,1\}^n, \exists z \in \{0,1\}^{nk(t)}, w = \mathrm{DP}_{k(t)}(x; z) \right] \\
&\le |L_t \cap \{0,1\}^n| \cdot 2^{nk(t)} \cdot 2^{-nk(t)-k(t)} \le A(1^t) \cdot 2^{-k(t)} = 2^{-d \log t} = t^{-d}.
\end{aligned} \tag{6}$$

We present a randomized algorithm B' that solves the promise problem Π defined in Theorem 4.2. On input $(x, 1^t)$, the algorithm B' lets $n := |x|$ and picks $z \sim \{0,1\}^{nk(t)}$ randomly, and accepts if and only if $B(\mathrm{DP}_{k(t)}(x; z), 1^{\langle n,t \rangle}) = 1$. Let $B'(x, 1^t; z)$ denote the output of this algorithm; that is, $B'(x, 1^t; z) = B(\mathrm{DP}_{k(t)}(x; z), 1^{\langle n,t \rangle})$. We claim the correctness of B' below. Let p_L be the polynomial of Definition 4.1.

Claim 4.7. *For all large $t \in \mathbb{N}$ and every $n \le p_L(t)$, the following hold for every $x \in \{0,1\}^n$.*

1. *If $x \in L_t$ ($\Leftrightarrow (x, 1^t) \in \Pi_{\text{Yes}}$), then $\Pr_z[B'(x, 1^t; z) = 1] = 1$.*
2. *If $(x, 1^t) \in \Pi_{\text{No}}$, then $\Pr_z[B'(x, 1^t; z) = 1] < 1 - \langle n,t \rangle^{-c'}/4$.*

Proof. Assume that $x \in L_t$. By the definition of L', we have $(\mathrm{DP}_{k(t)}(x; z), 1^{\langle n,t \rangle}) \in L'$ for every z. Since B is a one-sided-error algorithm that does not err on Yes instances, we obtain $1 = B(\mathrm{DP}_{k(t)}(x; z), 1^{\langle n,t \rangle}) = B'(x, 1^t; z)$. This completes the proof of Item 1.

To prove the contrapositive of Item 2, assume that $\Pr_z[B'(x, 1^t; z) = 1] \ge 1 - \langle n,t \rangle^{-c'}/4$; that is,

$$\Pr_z \left[B\left(\mathrm{DP}_{k(t)}(x; z), 1^{\langle n,t \rangle} \right) = 1 \right] \ge 1 - \langle n,t \rangle^{-c'}/4. \tag{7}$$

On the other hand,

$$\begin{aligned}\Pr_w\left[B(w,1^{\langle n,t\rangle})=1\right] &\le \Pr_w\left[L'(w,1^{\langle n,t\rangle})=1\right]+\Pr_w\left[B(w,1^{\langle n,t\rangle})\neq L'(w,1^{\langle n,t\rangle})\right]\\ &\le t^{-d}+1-\langle n,t\rangle^{-c'},\end{aligned}\tag{8}$$

where the last inequality uses Eq. (6). Choosing the constant d large enough (depending on p_L and c'), we have $t^{-d}+1-\langle n,t\rangle^{-c'}\le 1-\langle n,t\rangle^{-c'}/2$ for all large t. By Eqs. (7) and (8), we obtain

$$\Pr_z\left[B\Big(\mathrm{DP}_{k(t)}(x;z),1^{\langle n,t\rangle}\Big)=1\right]-\Pr_w\left[B(w,1^{\langle n,t\rangle})=1\right]\ge\langle n,t\rangle^{-c'}/4.$$

This means that $B(\text{-},1^{\langle n,t\rangle})$ ϵ-distinguishes the output distribution of $\mathrm{DP}_{k(t)}(x;\text{-})$ from the uniform distribution for $\epsilon:=\langle n,t\rangle^{-c'}/4$. By Theorem 3.12, we obtain $\mathrm{K}^{q(t)}(x)\le k(t)+\log q(t)$ for some polynomial q. Since $k(t)=\log A(1^t)+d\log t\le\log|L_t|+2+d\log t$, letting $p(t):=4q(t)t^d$, we obtain

$$\mathrm{K}^{p(t)}(x)\le\mathrm{K}^{q(t)}(x)\le\log|L_t|+2+d\log t+\log q(t)=\log|L_t|+\log p(t),$$

which means that $(x,1^t)\notin\Pi_{\mathrm{No}}$. $\diamond$

It follows from Claim 4.7 that $\Pi\in$ pr-coRP. Using Lemma 3.4, we conclude that $\Pi\in$ pr-BPP $=$ pr-P. $\square$

5 Weak Symmetry of Information

In this section, we prove the weak symmetry of information for time-bounded Kolmogorov complexity. The proof relies on the following worst-case-to-average-case connection of $\mathrm{Gap}(\mathrm{K}^A \text{ vs } \mathrm{K})$.

Lemma 5.1 ([Hir20b, Hir20a]). *For any oracle A, if $\mathsf{coNP}^A\times\{\mathcal{U},\mathcal{T}\}\subseteq\mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ for some constant c, then $\mathrm{Gap}(\mathrm{K}^A \text{ vs } \mathrm{K})\in$ pr-P.*

Proof. For completeness, we present a simple proof based on the algorithmic language compression theorem.[21] Consider an ensemble $L=\left\{L_{\langle t,s\rangle}\right\}_{t,s\in\mathbb{N}}$ of languages defined as

$$L_{\langle t,s\rangle}:=\left\{\,x\in\{0,1\}^*\;\middle|\;\mathrm{K}^{t,A}(x)\le s\,\right\}.$$

Observe that $|L_{\langle t,s\rangle}|\le 2^{s+1}$ by Fact 3.7 and $L\in\mathsf{NP}^A$. Applying Theorem 4.2 to L, we obtain a polynomial-time algorithm that solves the promise problem $(\Pi_{\mathrm{Yes}},\Pi_{\mathrm{No}})$ defined as

$$\begin{aligned}\Pi_{\mathrm{Yes}}&:=L=\left\{\,(x,1^{\langle t,s\rangle})\;\middle|\;\mathrm{K}^{t,A}(x)\le s\,\right\},\\ \Pi_{\mathrm{No}}&:=\left\{\,(x,1^{\langle t,s\rangle})\;\middle|\;\mathrm{K}^{p(t+s)}(x)>s+1+\log p(t+s)\,\right\}\end{aligned}$$

for some polynomial p. $\mathrm{Gap}_\tau(\mathrm{K}^A \text{ vs } \mathrm{K})$ is reducible to this promise problem via the identity map for some polynomial τ. $\square$

[21] To compare this proof with that of [Hir20a], the proof of [Hir20a] uses a padding argument specific to $\mathrm{Gap}(\mathrm{K}^A \text{ vs } \mathrm{K})$ in order to make a reduction error-tolerant; the proof presented here bypasses the padding argument using the notion of parameterized uniform distribution (Definition 3.5).

We are now ready to prove the weak symmetry of information.

Theorem 5.2 (Weak Symmetry of Information). *If* $\mathsf{coNP} \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ *for some constant* c*, then there exist polynomials* p_0 *and* p_{w} *such that, for any* $n, m \in \mathbb{N}$*, any* $t \geq p_0(nm)$*, any* $\epsilon > 0$*, and any* $x \in \{0,1\}^n$*,*

$$\Pr_{w \sim \{0,1\}^m}\left[\mathrm{K}^t(xw) \geq \mathrm{K}^{p_{\mathrm{w}}(t/\epsilon)}(x) + m - \log p_{\mathrm{w}}(t/\epsilon)\right] \geq 1 - \epsilon.$$

Proof. Using Lemma 5.1, we take a polynomial-time algorithm A that solves $\mathrm{Gap}_\tau\mathrm{MINKT}$ for some polynomial τ. Let k be a parameter chosen later. Take the k-wise direct product generator $\mathrm{DP}_k \colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{d+k}$, where $d = nk$. Let $s := d + k + m - \log \tau(d + k + m + 2t) - \log(2/\epsilon) - 1$. We prove the theorem by analyzing the behavior of $A(\mathrm{DP}_k(x; z) \cdot w, 1^{2t}, 1^s)$ for random choices of $w \sim \{0,1\}^m$ and $z \sim \{0,1\}^d$.

If A outputs 1, then the input is not a No instance of $\mathrm{Gap}_\tau\mathrm{MINKT}$; thus,

$$\Pr_{\substack{\omega \sim \{0,1\}^{d+k} \\ w \sim \{0,1\}^m}}\left[A(\omega \cdot w, 1^{2t}, 1^s) = 1\right] \leq \Pr_{\omega, w}\left[\mathrm{K}(\omega \cdot w) \leq s + \log \tau(d + k + m + 2t)\right] \leq \epsilon/2,$$

where the last inequality follows from Fact 3.7.

Assume that $\Pr_{z,w}\left[A(\mathrm{DP}_k(x; z) \cdot w, 1^{2t}, 1^s) = 1\right] \geq \epsilon$. These two inequalities mean that a randomized circuit D defined as $D(\omega; w) := A(\omega \cdot w, 1^{2t}, 1^s)$ can $\epsilon/2$-distinguish $\mathrm{DP}_k(x; \text{-})$ from the uniform distribution. By Theorem 3.12, there exists a polynomial p_{DP} such that $\mathrm{K}^{p_{\mathrm{DP}}(t/\epsilon)}(x) \leq k + \log p_{\mathrm{DP}}(t/\epsilon)$.

We now choose $k := \mathrm{K}^{p_{\mathrm{DP}}(t/\epsilon)}(x) - \log p_{\mathrm{DP}}(t/\epsilon) - 1$ so that this inequality does not hold. By the contrapositive of the argument above, we obtain $\Pr_{z,w}\left[A(\mathrm{DP}_k(x; z) \cdot w, 1^{2t}, 1^s) = 1\right] < \epsilon$.

This means that, with probability at least $1 - \epsilon$ over the choice of z and w, $A(\mathrm{DP}_k(x; z) \cdot w, 1^{2t}, 1^s) = 0$ holds. Under this event, $(\mathrm{DP}_k(x; z) \cdot w, 1^{2t}, 1^s)$ is not a Yes instance of $\mathrm{Gap}_\tau\mathrm{MINKT}$; that is,

$$\mathrm{K}^{2t}(\mathrm{DP}_k(x; z) \cdot w) > s.$$

Observe that $\mathrm{DP}_k(x; z) \cdot w$ can be described using $m, d \in \mathbb{N}$, $z \in \{0,1\}^d$, and $\mathrm{K}^t(xw)$ bits for describing xw; thus,

$$\mathrm{K}^{2t}(\mathrm{DP}_k(x; z) \cdot w) \leq \mathrm{K}^t(xw) + d + O(\log dm)$$

for a sufficiently large $t \geq p_0(nm)$. Combining these two inequalities, we obtain

$$\begin{aligned}
\mathrm{K}^t(xw) &> s - d - O(\log dm) \\
&\geq k + m - O(\log(\tau(d + k + m + 2t) dm/\epsilon)) \\
&\geq \mathrm{K}^{p_{\mathrm{w}}(t/\epsilon)}(x) + m - \log p_{\mathrm{w}}(t/\epsilon),
\end{aligned}$$

where p_{w} is some large polynomial. □

6 Fast Algorithms from Universal Heuristic Schemes

Recall the notion of time-bounded computational depth and universal heuristic scheme.

Definition 6.1 (Time-Bounded Computational Depth)**.** *For an oracle A, time bounds $s \in \mathbb{N}$ and $t \in \mathbb{N} \cup \{\infty\}$, and a string $x \in \{0,1\}^*$, the A-*oracle (s,t)-time-bounded computational depth *of x is defined as*

$$\mathrm{cd}^{s,t,A}(x) := \mathrm{K}^{s,A}(x) - \mathrm{K}^{t}(x).$$

We omit the superscript A if $A = \varnothing$, and the superscript t if $t = \infty$.

Definition 6.2 (Universal Heuristic Scheme)**.** *A* ($\mathsf{Avg_P P}$-)universal heuristic scheme *for a language L is a pair (S,C) of polynomial-time algorithms such that, for some polynomial p, for any $n \in \mathbb{N}$, any $t \geq p(n)$, and any $x \in \{0,1\}^n$,*

1. *if $\mathrm{cd}^{t,p(t)}(x) \leq k$, then $C(x, 1^t, 1^k) = 1$, and*
2. *if $C(x, 1^t, 1^k) = 1$, then $S(x, 1^t, 1^{2^k}) = L(x)$.*

S and C are referred to as a solver *and a* checker*, respectively.*

An important property of a universal heuristic scheme is that the existence of a universal heuristic scheme for a language L implies a fast worst-case algorithm for L. Any language L with low time-bounded Kolmogorov complexity admits a faster worst-case algorithm.

Theorem 6.3. *Let $s\colon \mathbb{N} \to \mathbb{N}$ be an efficiently computable function such that $\log n < s(n) < O(n)$, and let p_0 be a polynomial. Let L be a language such that $\mathrm{K}^{p_0(|x|)}(x) \leq s(|x|)$ for every $x \in L$. If there exists a universal heuristic scheme (S,C) for a language L, then $L \in \mathsf{DTIME}\big(2^{O(s(n)/\log(s(n)/\log n))}\big)$.*

Proof. Let $p \geq p_0$ be a sufficiently large polynomial that satisfies the definition of a universal heuristic scheme (S,C) (i.e., Definition 6.2). We recursively define $p^i(n)$ so that $p^0(n) = n$ and $p^{i+1}(n) = p(p^i(n))$ for every $n \in \mathbb{N}$.

We present an algorithm A that solves L on inputs of length n. Let I be a parameter chosen later, and let $k := s(n)/I$. The algorithm A finds $i \in [I]$ such that $C(x, 1^{p^i(n)}, 1^k) = 1$; if such an i is not found, A rejects and halts. A accepts if and only if $S(x, 1^{p^i(n)}, 1^{2^k}) = 1$.

We claim the correctness of the algorithm A. Note that A rejects every $x \notin L$ because of the correctness of (S,C). Thus, it suffices to prove that, for every $x \in L$, there exists $i \in [I]$ such that $C(x, 1^{p^i(n)}, 1^k) = 1$. Consider the following telescoping sum:

$$\mathrm{K}^{p(n)}(x) - \mathrm{K}^{p^{I+1}(n)}(x) = \mathrm{cd}^{p^1(n),p^2(n)}(x) + \mathrm{cd}^{p^2(n),p^3(n)}(x) + \cdots + \mathrm{cd}^{p^I(n),p^{I+1}(n)}(x).$$

This sum is at most $\mathrm{K}^{p(n)}(x) \leq s(n)$. We thus obtain

$$I \cdot \min\left\{ \mathrm{cd}^{p^i(n),p^{i+1}(n)}(x) \;\middle|\; i \in [I] \right\} \leq \mathrm{K}^{p(n)}(x) - \mathrm{K}^{p^{I+1}(n)}(x) \leq s(n),$$

from which it follows that there exists $i \in [I]$ such that $\mathrm{cd}^{p^i(n),p^{i+1}(n)}(x) \leq s(n)/I = k$; we conclude that $C(x, 1^{p^i(n)}, 1^k) = 1$ using the property of the checker C.

We claim that the running time of A is bounded by $2^{O(s(n)/\log(s(n)/\log n))}$. Let $c > 1$ be a constant such that $p(n) \leq n^c$ for all large n. By induction, $p^i(n) \leq n^{c^i}$ for all large n and every i. We define I to be $\max\{1, \log_c(s(n)/\log n) - \log_c \log(s(n)/\log n) - 1\}$; we then obtain $p^{I+1}(n) \leq 2^{O(s(n)/\log(s(n)/\log n))}$ for all large n. For every $i \in [I+1]$, the running time of $S(x, 1^{p^i(n)}, 1^{2^k})$ is bounded by $\mathsf{poly}\big(n, 2^{s(n)/\log(s(n)/\log n)}, 2^{s(n)/I}\big) = 2^{O(s(n)/\log(s(n)/\log n))}$. □

We present a couple of corollaries of Theorem 6.3. Using the trivial upper bound of Kolmogorov complexity, we obtain the following corollary.

Corollary 6.4. *If there exists a universal heuristic scheme* (S, C) *for a language* L*, then* $L \in \mathsf{DTIME}\big(2^{O(n/\log n)}\big)$.

Proof. For some polynomial p_0, for every string x of length n, we have $\mathrm{K}^{p_0(n)}(x) \le n + O(1)$ because any string $x \in \{0,1\}^n$ can be described by x itself. Applying Theorem 6.3 to $s(n) := n + O(1)$, we obtain $L \in \mathsf{DTIME}\big(2^{O(n/\log(n/\log n))}\big) = \mathsf{DTIME}\big(2^{O(n/\log n)}\big)$. $\square$

A padding argument produces a language with small Kolmogorov complexity:

Corollary 6.5. *Let* $t \colon \mathbb{N} \to \mathbb{N}$ *be a time-constructible function such that* $n \le t(n) < 2^n$*. Let* $\mathfrak{C}$ *be a complexity class and let* $\mathfrak{C}\text{-}\mathsf{TIME}(t(n))$ *denote its* $t(n)$*-time version.*[22] *If every language in* $\mathfrak{C}$ *admits a universal heuristic scheme, then* $\mathfrak{C}\text{-}\mathsf{TIME}(t(n)) \subseteq \mathsf{DTIME}\big(2^{O(n/\log(n/\log t(n)))}\big)$.

Proof. Take any language $L \in \mathfrak{C}\text{-}\mathsf{TIME}(t(n))$. Let $L' := \left\{ x01^{t(|x|)} \;\middle|\; x \in L \right\}$. Since $L' \in \mathfrak{C}$, L' admits a universal heuristic scheme. Moreover, for every $n \in \mathbb{N}$ and every $y \in L' \cap \{0,1\}^{n+1+t(n)}$, we have $\mathrm{K}^{p(t(n))}(y) \le n + O(1)$, where $p(n)$ is some polynomial. Using Theorem 6.3, we obtain an algorithm that solves L' on inputs of length $n + 1 + t(n)$ in time $2^{O(n/\log(n/\log t(n)))}$. $\square$

More generally, any sparse language has small Kolmogorov complexity. For a function $s \colon \mathbb{N} \to \mathbb{N}$, we say that a language L is s-sparse if $|L \cap \{0,1\}^n| \le 2^{s(n)}$ for every $n \in \mathbb{N}$.

Corollary 6.6. *Let* $s \colon \mathbb{N} \to \mathbb{N}$ *be an efficiently computable function such that* $\log n < s(n) < O(n)$*. Assume that* $\mathsf{coNP}^A \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ *for some constant* c *and some oracle* A*. Let* $L \in \mathsf{NP}^A$ *be an* s*-sparse language. If there exists a universal heuristic scheme* (S, C) *for a language* L*, then* $L \in \mathsf{DTIME}\big(2^{O(s(n)/\log(s(n)/\log n))}\big)$.

Proof. By Corollary 4.4, there exists a polynomial p such that $\mathrm{K}^{p(n)}(x) \le \log|L \cap \{0,1\}^n| + \log p(n) \le s(n) + \log p(n)$. The result readily follows from Theorem 6.3. $\square$

7 Universal Heuristic Scheme for PH

In this section, we construct a universal heuristic scheme for PH under the assumption that PH is easy on average. The whole section is devoted to proving the following.

Theorem 7.1. *For every constant* $\ell \in \mathbb{N}$*, if* $\mathsf{co}\Sigma^{\mathrm{p}}_{\ell+1} \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ *for some constant* c*, then every language in* $\mathsf{BPP}^{\Sigma^{\mathrm{p}}_\ell}$ *admits a universal heuristic scheme.*

In fact, the following stronger result holds.

Lemma 7.2. *For every constant* $\ell \in \mathbb{N}$*, if* $\mathrm{Gap}(\mathrm{K}^{\Sigma^{\mathrm{p}}_\ell} \text{ vs } \mathrm{K}) \in \text{pr-}\mathsf{P}$*, then every language in* $\mathsf{BPP}^{\Sigma^{\mathrm{p}}_\ell}$ *admits a universal heuristic scheme.*

[22] We can define $\mathfrak{C}\text{-}\mathsf{TIME}(t(n))$ as the class of languages L such that $L' := \left\{ x01^{t(|x|)} \;\middle|\; x \in L \right\} \in \mathfrak{C}$. This is essentially consistent with the standard definition of, for example, $\mathsf{NTIME}(t(n))$ in the sense that $\mathsf{NP}\text{-}\mathsf{TIME}(t(n)^{O(1)}) = \mathsf{NTIME}(t(n)^{O(1)})$.

Proof of (Lemma 7.2 ⇒ Theorem 7.1). Under the assumption that $\mathsf{co}\Sigma^{\mathrm{p}}_{\ell+1} \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{Avg}^{1}_{1-n^{-c}}\mathsf{P}$, Lemma 5.1 implies that $\mathrm{Gap}(\mathrm{K}^{\Sigma^{\mathrm{p}}_{\ell}} \text{ vs } \mathrm{K}) \in \text{pr-}\mathsf{P}$. □

Recall that the complexity of P^{NP} is characterized as the complexity of finding the lexicographically maximum certificate.

Lemma 7.3 (Implicit in [Kre88]). *Let A be an oracle. For every language $L \in \mathsf{P}^{\mathsf{NP}^{A}}$, there exist an A-oracle polynomial-time algorithm V^{A} and polynomials p_{y}, m such that, for every $x \in \{0,1\}^{*}$, $L(x)$ is equal to the $m(|x|)$-th bit of y_x, where y_x is the lexicographically maximum string $y \in \{0,1\}^{p_{\mathrm{y}}(|x|)}$ such that $V^{A}(x,y) = 1$. The algorithm V^{A} is referred to as a $\mathsf{P}^{\mathsf{NP}^{A}}$-type verifier.*

Proof Sketch. Let $B \in \mathsf{NP}^{A}$ and M be a polynomial-time oracle machine such that $M^{B}(x) = L(x)$ for every $x \in \{0,1\}^{*}$. Let m be a polynomial such that the number of queries M makes on inputs of length n is exactly $m(n)$. We define an A-oracle polynomial-time algorithm V^{A} as follows. Fix any input x of length n, and let $m := m(n)$. V^{A} takes (x,y) as input and regard y as a string $(a_1, \ldots, a_m, b, w_1, \ldots, w_m)$, where $a_i \in \{0,1\}$ for every $i \in [m]$ and $b \in \{0,1\}$. For every $i \in [m]$, V^{A} computes the i-th query q_i assuming that the j-th answer from the oracle B is a_j for every $j < i$. V^{A} accepts if and only if b is equal to the output of $M^{(\text{-})}(x)$ given the answers $(a_1, \ldots, a_m)$ from the oracle, and for every $i \in [m]$, $a_i = 1$ implies that w_i is an NP^{A}-certificate for $q_i \in B$. It is easy to see that the lexicographically maximum string y_x is well defined and that the first m bits of y_x are equal to $B(q_1) \cdots B(q_m)$, in which case the $(m+1)$-th bit is equal to $L(x)$. □

Proof of Lemma 7.2. We show that every language $L \in \mathsf{P}^{\Sigma^{\mathrm{p}}_{\ell}}$ admits a universal heuristic scheme. This is sufficient, as we will show in Lemma 7.10 that a universal heuristic scheme for $\mathsf{P}^{\Sigma^{\mathrm{p}}_{\ell}}$ can be converted into a universal heuristic scheme for $\mathsf{BPP}^{\Sigma^{\mathrm{p}}_{\ell}}$.

Claim 7.4. *For every constant $\ell \in \mathbb{N}$, if $\mathrm{Gap}(\mathrm{K}^{\Sigma^{\mathrm{p}}_{\ell}} \text{ vs } \mathrm{K}) \in \text{pr-}\mathsf{P}$, then every language $L \in \mathsf{P}^{\Sigma^{\mathrm{p}}_{\ell}}$ admits a universal heuristic scheme.*

We prove this claim by induction on ℓ. If $\ell = 0$, the claim is trivial; thus, we assume $\ell \geq 1$. We fix a canonical $\Sigma^{\mathrm{p}}_{\ell-1}$-complete problem A. Applying Lemma 7.3 to the language $L \in \mathsf{P}^{\mathsf{NP}^{A}} = \mathsf{P}^{\Sigma^{\mathrm{p}}_{\ell}}$, let V^{A} be a $\mathsf{P}^{\mathsf{NP}^{A}}$-type verifier for $L \in \mathsf{P}^{\mathsf{NP}^{A}}$. For each $x \in \{0,1\}^{*}$, let y_x denote the lexicographically maximum string $y \in \{0,1\}^{p_{\mathrm{y}}(|x|)}$ such that $V^{A}(x,y) = 1$.

Let $L_V := \left\{ (x,y) \;\middle|\; V^{A}(x,y) = 1 \right\} \in \mathsf{P}^{A}$ denote the language accepted by V^{A}. By the induction hypothesis, there exists a universal heuristic scheme (S_0, C_0) for $L_V \in \mathsf{P}^{A} = \mathsf{P}^{\Sigma^{\mathrm{p}}_{\ell-1}}$. In particular, there exists a polynomial p such that, for every $n \in \mathbb{N}$, every $x \in \{0,1\}^{n}$, every $t \geq p(n)$, if $\mathrm{cd}^{t,p(t)}(x) \leq k$, then $C_0(x, y, 1^{t}) = 1$ and $S_0(x, y, 1^{t}, 1^{2^{k}}) = L_V(x,y) = V^{A}(x,y)$.

Take a canonical $\Sigma^{\mathrm{p}}_{\ell}$-complete problem B.[23] Using a standard search-to-decision procedure, given $x \in \{0,1\}^{*}$ as input and oracle access to B, one can compute y_x in polynomial time. In particular, there exists a polynomial $p_0 \geq p$ such that, for every $x \in \{0,1\}^{*}$ and every $t \geq p_0(|x|)$,

$$\mathrm{K}^{2t,B}(x, y_x) \leq \mathrm{K}^{t}(x) + O(\log n). \tag{9}$$

Using the assumption, let M_{K} be a polynomial-time algorithm that solves $\mathrm{Gap}_{\tau}(\mathrm{K}^{B} \text{ vs } \mathrm{K})$ for some polynomial τ. Let $p_{\mathrm{K}}(t) := \tau(2t)$. Below, we define a universal heuristic scheme (S, C) for L.

[23] Alternatively, $B \in \Sigma^{\mathrm{p}}_{\ell}$ can be defined as the set of strings (x, y_0) such that $V^{A}(x,y) = 1$ for some y that has y_0 as prefix. This simplifies the proof of Eq. (9).

We define a checker C. Let q be a large polynomial chosen later. Fix any input $(x, 1^t, 1^k)$ such that $t \geq p_0(|x|)$. Using M_{K} and Fact 3.8, the checker C computes values s and v such that

$$\mathrm{K}^{p_{\mathrm{K}}(t)}(x) \leq s \leq \mathrm{K}^{t,B}(x) + \log p_{\mathrm{K}}(t),$$
$$\mathrm{K}^{p_{\mathrm{K}}(q(t))}(x) \leq v \leq \mathrm{K}^{q(t),B}(x) + \log p_{\mathrm{K}}(q(t)).$$

The checker C accepts if and only if $s - v \leq k + \log p_{\mathrm{K}}(t)$.

We prove two properties of the checker C in the following two claims.

Claim 7.5. *If* $\mathrm{cd}^{t,p_{\mathrm{K}}(q(t)),B}(x) \leq k$*, then* $C(x, 1^t, 1^k) = 1$.

Proof. By the definition of s and v, we obtain

$$s - v \leq \mathrm{K}^{t,B}(x) + \log p_{\mathrm{K}}(t) - \mathrm{K}^{p_{\mathrm{K}}(q(t))}(x) = \mathrm{cd}^{t,p_{\mathrm{K}}(q(t)),B}(x) + \log p_{\mathrm{K}}(t) \leq k + \log p_{\mathrm{K}}(t).$$

◇

Claim 7.6. *There exists a polynomial* p_C *such that*

$$C(x, 1^t, 1^k) = 1 \implies \mathrm{cd}^{p_{\mathrm{K}}(t),q(t)}(x) \leq k + \log p_C(t).$$

Proof. Observe that

$$k + \log p_{\mathrm{K}}(t) \geq s - v \geq \mathrm{K}^{p_{\mathrm{K}}(t)}(x) - \mathrm{K}^{q(t),B}(x) - \log p_{\mathrm{K}}(q(t)) \geq \mathrm{cd}^{p_{\mathrm{K}}(t),q(t)}(x) - \log p_{\mathrm{K}}(q(t)).$$

Choosing $\log p_C(t) := \log p_{\mathrm{K}}(t) + \log p_{\mathrm{K}}(q(t))$, we obtain the claim. ◇

Now, we define a solver S. Fix an input $(x, 1^t, 1^{2^k})$, and let $n := |x|$. We assume that $t \geq p_0(n)$ and $C(x, 1^t, 1^k) = 1$. Using M_{K} and Fact 3.8, the solver S computes a value s' such that

$$\mathrm{K}^{p_{\mathrm{K}}^2(2t)}(x) \leq s' \leq \mathrm{K}^{p_{\mathrm{K}}(2t),B}(x) + \log p_{\mathrm{K}}(2t). \tag{10}$$

Let s'', t', k', k_0, and t_0 be some parameters chosen later. The solver S defines a function $D \colon \{0,1\}^d \to \{0,1\}^{d+k'}$ so that $D(w) := M_{\mathrm{K}}\Big((x, w), 1^{t'}, 1^{s''}\Big)$, where $d := k' \cdot p_{\mathrm{y}}(n)$. Applying Theorem 3.12 to D, the solver S computes the set Y of all the strings $y \in \{0,1\}^{p_{\mathrm{y}}(n)}$ such that $\mathrm{K}^{p_{\mathrm{DP}}(t)}(y \mid D) \leq k' + \log p_{\mathrm{DP}}(t)$, and then computes

$$y^* := \max\left\{ y \in Y \;\middle|\; C_0(x, y, 1^{t_0}, 1^{k_0}) = S_0(x, y, 1^{t_0}, 1^{2^{k_0}}) = 1 \right\}.$$

The solver S accepts if and only if the $m(n)$-th bit of y^* is equal to 1, where m is the polynomial of Lemma 7.3.

We prove the correctness of S by the two claims given below.

Claim 7.7. *If* $C(x, 1^t, 1^k) = 1$*, then* $y_x \in Y$.

Proof. It suffices to claim that D $\frac{1}{2}$-distinguishes the output distribution of $\mathrm{DP}_{k'}(y_x; \text{-})$ from the uniform distribution.

On one hand, observe that

$$\begin{aligned}\mathrm{K}^{p_{\mathrm{K}}^2(2t),B}(x, \mathrm{DP}_{k'}(y_x; z)) &\leq \mathrm{K}^{p_{\mathrm{K}}^2(t)}(x) + |z| + O(\log n)\\ &\leq s' + d + \log p_1(t)\end{aligned}$$

for some large polynomial p_1, where the first inequality is from Eq. (9). This means that

$$\left((x, \mathrm{DP}_{k'}(y_x; z)), 1^{t'}, 1^{s''}\right)$$

is a YES instance of $\mathrm{Gap}_\tau(\mathrm{K}^B \text{ vs } \mathrm{K})$ for $t' := p_{\mathrm{K}}^2(2t)$ and $s'' := s' + d + \log p_1(t)$. Thus, $D(\mathrm{DP}_{k'}(y_x; z)) = 1$ for every $z \in \{0,1\}^d$.

On the other hand, observe that

$$\begin{aligned} k &\geq \mathrm{cd}^{p_{\mathrm{K}}(t), q(t)}(x) - \log p_C(t) && \text{(by Claim 7.6)} \\ &\geq s' - \log p_{\mathrm{K}}(2t) - \mathrm{K}^{q(t)}(x) - \log p_C(t) && \text{(by Eq. (10))}. \end{aligned}$$

By Theorem 5.2, with probability at least $\frac{1}{2}$ over a random choice of $w \sim \{0,1\}^{d+k'}$,

$$\begin{aligned} \mathrm{K}^{p_{\mathrm{K}}^3(2t)}(x, w) &\geq \mathrm{K}^{p_{\mathrm{w}}(p_{\mathrm{K}}^3(2t))}(x) + d + k' - \log p_{\mathrm{w}}(p_{\mathrm{K}}^3(2t)) \\ &> s' + d + \log p_1(t) + \log p_{\mathrm{K}}^3(2t), \end{aligned}$$

where the last inequality holds by letting $q(t) \geq p_{\mathrm{w}}(p_{\mathrm{K}}^3(2t))$ and choosing

$$\begin{aligned} k' &:= k + \log p_{\mathrm{K}}(2t) + \log p_C(t) + \log p_1(t) + \log p_{\mathrm{K}}^3(2t) + \log p_{\mathrm{w}}(p_{\mathrm{K}}^3(2t)) + 1 \\ &> s' - \mathrm{K}^{q(t)}(x) + \log p_1(t) + \log p_{\mathrm{K}}^3(2t) + \log p_{\mathrm{w}}(p_{\mathrm{K}}^3(2t)). \end{aligned}$$

Under the event, $\left((x, w), 1^{t'}, 1^{s''}\right)$ is a NO instance of $\mathrm{Gap}_\tau(\mathrm{K}^B \text{ vs } \mathrm{K})$; thus, $\Pr_w[D(w) = 1] \leq \frac{1}{2}$.

We conclude that D $\frac{1}{2}$-distinguishes $\mathrm{DP}_{k'}(y_x; \text{-})$ from the uniform distribution. By Theorem 3.12, we obtain $y_x \in Y$. $\diamond$

Claim 7.8. *If $C(x, 1^t, 1^k) = 1$, then $y^* = y_x$.*

Proof. We first claim that $y^* \leq y_x$. By the definition, we have $C_0(x, y^*, 1^{t_0}, 1^{k_0}) = S_0(x, y^*, 1^{t_0}, 1^{2^{k_0}}) = 1$. By the property of the universal heuristic scheme (S_0, C_0), we have $L_V(x, y^*) = V^A(x, y^*) = 1$ and thus $y^* \leq y_x$.

In order to prove $y^* \geq y_x$, it suffices to prove $\mathrm{cd}^{t_0, p(t_0)}(x, y_x) \leq k_0$. Indeed, by the definition of y_x, $V^A(x, y_x) = 1$; thus, $S_0(x, y_x, 1^{t_0}, 1^{2^{k_0}}) = V^A(x, y_x) = 1$ under the assumption that $C_0(x, y_x, 1^{t_0}, 1^{k_0}) = 1$, which is true if $\mathrm{cd}^{t_0, p(t_0)}(x, y_x) \leq k_0$.

We prove $\mathrm{cd}^{t_0, p(t_0)}(x, y_x) \leq k_0$. Since $y_x \in Y$ can be described using a self-delimiting encoding of D and $\mathrm{K}^{p_{\mathrm{DP}}(t)}(y_x \mid D) \leq k' + \log p_{\mathrm{DP}}(t)$ bits of information, we have

$$\mathrm{K}^{2p_{\mathrm{K}}(t) + 2p_{\mathrm{DP}}(t)}(x, y_x) \leq \mathrm{K}^{p_{\mathrm{K}}(t)}(x) + k' + \log p_{\mathrm{DP}}(t) + O(\log t).$$

Let $t_0 := 2p_{\mathrm{K}}(t) + 2p_{\mathrm{DP}}(t)$. Then we obtain

$$\begin{aligned} \mathrm{cd}^{t_0, p(t_0)}(x, y_x) &\leq \mathrm{K}^{p_{\mathrm{K}}(t)}(x) + k' + O(\log t) - \mathrm{K}^{2p(t_0)}(x) && (\text{since } \mathrm{K}^{p(t_0)}(x, y_x) \geq \mathrm{K}^{2p(t_0)}(x)) \\ &\leq k' + \mathrm{cd}^{p_{\mathrm{K}}(t), q(t)}(x) + O(\log t) && (\text{by letting } q(t) \geq 2p(t_0)) \\ &\leq k' + k + \log p_C(t) + O(\log t) && \text{(by Claim 7.6)} \\ &=: k_0. \end{aligned}$$

$\diamond$

The correctness of the solver S follows from the claims above: If $C(x, 1^t, 1^k) = 1$, then the solver S computes $y^* = y_x$; thus, $S(x, 1^t, 1^{2^k}) = L(x)$ by Lemma 7.3. Finally, we note that $k_0 = 2k + O(\log t)$; thus, the solver S runs in time $\mathsf{poly}(n, t, 2^k)$. $\square$

It remains to convert a universal heuristic scheme for $\mathsf{P}^{\mathsf{\Sigma}_\ell^{\mathrm{p}}}$ to that for $\mathsf{BPP}^{\mathsf{\Sigma}_\ell^{\mathrm{p}}} = \mathsf{BP}\cdot\mathsf{P}^{\mathsf{\Sigma}_\ell^{\mathrm{p}}}$. Recall the notion of BP-operator.

Definition 7.9 (BP-operator)**.** *For a complexity class $\mathfrak{C}$, let $\mathsf{BP}\cdot\mathfrak{C}$ denote the class of languages L such that there exist $L' \in \mathfrak{C}$ and a polynomial p such that, for every $x \in \{0,1\}^*$,*

$$\Pr_{r\sim\{0,1\}^{p(|x|)}}\left[L'(x,r) = L(x)\right] \geq \frac{8}{9}.$$

Lemma 7.10. *Let $\mathfrak{C}$ be any complexity class. Assume that $\mathsf{coNP}\times\{\mathcal{U},\mathcal{T}\} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ for some constant c. If any language in $\mathfrak{C}$ admits a universal heuristic scheme, then any language in $\mathsf{BP}\cdot\mathfrak{C}$ admits a universal heuristic scheme.*

Proof. Let $\epsilon := \frac{1}{9}$. Take an arbitrary language $L \in \mathsf{BP}\cdot\mathfrak{C}$, and let $L' \in \mathfrak{C}$ be a language such that, for every input x of length n,

$$\Pr_{r\sim\{0,1\}^{p(n)}}\left[L'(x,r) \neq L(x)\right] \leq \epsilon. \tag{11}$$

By the assumption, there exists a universal heuristic scheme (S, C) for L'.

By Theorem 5.2, there exist polynomials p_0 and p_{w} such that, for every $n \in \mathbb{N}$, $t \geq p_0(n)$, and $x \in \{0,1\}^n$, with probability at least $1-\epsilon$ over a random choice of $r \sim \{0,1\}^{p(n)}$,

$$\begin{aligned}
\mathrm{cd}^{2t,p(2t)}(x,r) &= \mathrm{K}^{2t}(x,r) - \mathrm{K}^{p(2t)}(x,r)\\
&\leq \left(\mathrm{K}^t(x) + |r| + O(\log n)\right) - \left(\mathrm{K}^{p_{\mathrm{w}}(p(2t))}(x) + |r| - \log p_{\mathrm{w}}(p(2t))\right)\\
&\leq \mathrm{cd}^{t,p_{\mathrm{w}}(p(2t))}(x) + O(\log t)\\
&\leq \mathrm{cd}^{t,p_1(t)}(x) + \log p_1(t),
\end{aligned}$$

where p_1 is some large polynomial. We define a checker C' so that $C'(x, 1^t, 1^k) = 1$ if and only if

$$\Pr_{z\sim\{0,1\}^{O(\log m)}}\left[C\Big((x, G_m(z)), 1^{2t}, 1^{k'}\Big) = 1\right] \geq 1 - 2\epsilon,$$

where $k' := k + \log p_1(t)$ and G_m is a pseudorandom generator of Lemma 3.4 for a sufficiently large $m \leq \mathsf{poly}(n,t,k)$. It follows from the argument above that, if $\mathrm{cd}^{t,p_1(t)}(x) \leq k$, then $C'(x,1^t,1^k) = 1$.

We define a solver S' so that $S'(x, 1^t, 1^{2^k})$ is equal to the majority of $S\Big((x, G_{m'}(z)), 1^{2t}, 1^{2^{k'}}\Big)$ over a choice of $z \in \{0,1\}^{O(\log m')}$ for a sufficiently large $m' \leq \mathsf{poly}(n,t,2^k)$. We claim that $C'(x,1^t,1^k) = 1$ implies $S'(x,1^t,1^{2^k}) = L(x)$. If $C'(x,1^t,1^k) = 1$, then by the property of the pseudorandom generator G_m, with probability at least $1-3\epsilon$ over a random choice of r, we have $C\Big((x,r),1^{2t},1^{k'}\Big) = 1$, in which case $S\Big((x,r),1^{2t},1^{2^{k'}}\Big) = L'(x,r)$. Using Eq. (11) and a union bound, we obtain

$$\Pr_r\left[S\Big((x,r),1^{2t},1^{2^{k'}}\Big) = L(x)\right] \geq 1 - 4\epsilon.$$

It follows from the security of the pseudorandom generator $G_{m'}$ that $S'(x,1^t,1^{2^k}) = L(x)$.

We conclude that (S', C') is a universal heuristic scheme for L. □

8 Approximately Size-Verifiable One-Way Function

The goal of this section is to construct a universal heuristic scheme for inverting an approximately size-verifiable one-way function in the worst case under the assumption that NP is easy on average. In Section 8.1, we introduce "size-verifiable NP", denoted by $\mathsf{NP}_{\mathsf{sv}}$, which is a decision version of inverting size-verifiable one-way functions. In Section 8.2, we construct a universal heuristic scheme for the search version of $\mathsf{NP}_{\mathsf{sv}}$. The definition of approximately size-verifiable one-way functions is given in Section 8.3.

8.1 Size-Verifiable NP

We introduce a notation for the size of certificates.

Definition 8.1 (NP-type verifier and the size of certificates)**.** *An algorithm V is said to be an* NP-type verifier *for a language L if there exists a function $t\colon \mathbb{N} \to \mathbb{N}$ such that, for every $x \in \{0,1\}^*$, $x \in L$ if and only if there exists $y \in \{0,1\}^{t(|x|)}$ such that $V(x,y) = 1$. For an NP-type verifier V, define $\sigma_V\colon \{0,1\}^* \to \mathbb{N} \cup \{-\infty\}$ as*

$$\sigma_V(x) := \log \# \left\{ y \in \{0,1\}^{p(|x|)} \;\middle|\; V(x,y) = 1 \right\}.$$

The notion of *size-verifiability* was introduced in the work of Akavia, Goldreich, Goldwasser, and Moshkovitz [AGGM06] in the context of one-way functions. Using the notion of size-verifiability, we introduce the subclass $\mathsf{NP}_{\mathsf{sv}}$ of NP.

Definition 8.2 ($\mathsf{NP}_{\mathsf{sv}}$; Size-Verifiable NP)**.** *For a function $t\colon \mathbb{N} \to \mathbb{N}$, we define $\mathsf{NTIME}_{\mathsf{sv}}(t(n))$ as the class of languages L such that there exists an algorithm V (referred to as* an $\mathsf{NP}_{\mathsf{sv}}$-type verifier*) satisfying the following.*

1. *V is an NP-type verifier for L; that is, for every $x \in \{0,1\}^*$, $x \in L$ if and only if there exists $y \in \{0,1\}^{t(|x|)}$ such that $V(x,y) = 1$.*

2. *V halts in time $O(t(n))$ on input $(x,y) \in \{0,1\}^n \times \{0,1\}^{t(n)}$ for every $n \in \mathbb{N}$.*

3. *Define a promise problem $\Pi^V = (\Pi^V_{\mathrm{YES}}, \Pi^V_{\mathrm{NO}})$ as follows:*

$$\begin{aligned}\Pi^V_{\mathrm{YES}} &:= \{\, (x,\ell) \in \{0,1\}^* \times \mathbb{N} \mid 0 \le \sigma_V(x) \le \ell \,\},\\ \Pi^V_{\mathrm{NO}} &:= \{\, (x,\ell) \in \{0,1\}^* \times \mathbb{N} \mid \sigma_V(x) > \ell + \log t(|x|) \,\}.\end{aligned}$$

   Then, $\Pi^V \in \text{pr-}\mathsf{AMTIME}(t(n))$.

We define $\mathsf{NP}_{\mathsf{sv}} := \bigcup_{c \in \mathbb{N}} \mathsf{NTIME}_{\mathsf{sv}}(n^c)$.

Remark 8.3. The promise problem Π^V of Definition 8.2 asks to approximate the number of certificates for x within a factor of $t(|x|)$ for every $x \in L$. Note that any input $x \notin L$ is not in the promise of Π^V; indeed, $(x,\ell) \notin \Pi^V_{\mathrm{YES}} \cup \Pi^V_{\mathrm{NO}}$ holds for every $\ell \in \mathbb{N}$ because $\sigma_V(x) = -\infty$.

Fact 8.4. $\mathsf{UP} \subseteq \mathsf{FewP} \subseteq \mathsf{NP}_{\mathsf{sv}} \subseteq \mathsf{NP}$.

Proof. We prove $\mathsf{FewP} \subseteq \mathsf{NP}_{\mathsf{sv}}$. Consider a language $L \in \mathsf{FewP}$. By the definition of FewP [AR88], there exist a polynomial p and an NP-type verifier V for L such that $\sigma_V(x) \le \log p(|x|)$ for any $x \in \{0,1\}^*$. Since $\sigma_V(x) \le \log p(|x|)$ holds for any x, the promise problem Π^V defined in Definition 8.2 satisfies $\Pi^V_{\mathrm{No}} = \varnothing$, and hence Π^V can be solved by a trivial algorithm that always outputs 1; thus, we obtain $L \in \mathsf{NP}_{\mathsf{sv}}$. $\square$

To construct a universal heuristic scheme for $\mathsf{NP}_{\mathsf{sv}}$, we will exploit the property that $\mathsf{NP}_{\mathsf{sv}}$ admits an AM procedure W that reduces the number of certificates.

Lemma 8.5. *For every language $L \in \mathsf{NP}_{\mathsf{sv}}$, there exist polynomial-time algorithms V and W and a polynomial p such that, for every string x of length n, the following hold.*

1. *$x \in L$ if and only if $V(x,y) = 1$ for some $y \in \{0,1\}^{p(n)}$.*
2. *If $W(x,y;y',r) = 1$ holds for some y' and some $r \in \{0,1\}^{p(n)}$, then $V(x,y) = 1$.*
3. *If $x \in L$, then with probability at least $\frac{1}{2}$ over the choice of $r \sim \{0,1\}^{p(n)}$, there exist strings $y, y' \in \{0,1\}^{p(n)}$ such that $W(x,y;y',r) = 1$.*
4. *With probability at least $1 - 2^{-2n}$ over the choice of $r \sim \{0,1\}^{p(n)}$,*
$$\# \left\{ y \in \{0,1\}^{p(n)} \;\middle|\; \exists y' \in \{0,1\}^{p(n)}, W(x,y;y',r) = 1 \right\} \le p(n).$$

The proof idea of Lemma 8.5 is to pick a k-wise independent hash h randomly and restrict the certificate that h maps to the all-0 string. Recall the standard construction of a k-wise independent hash family.

Lemma 8.6 (k-wise independent hash; cf. [Vad12, Corollary 3.34])**.** *For every $n, m, k \in \mathbb{N}$, there is a family of k-wise independent hash functions $\mathcal{H}_{n,m,k} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ such that choosing a random function from $\mathcal{H}_{n,m,k}$ takes $k \cdot \max\{n,m\}$ random bits, and evaluating a function from $\mathcal{H}_{n,m,k}$ takes time $\mathsf{poly}(n,m,k)$.*

We use the following concentration inequality for k-wise independent variables.

Lemma 8.7 ([SSS95])**.** *Let $X_1, \cdots, X_n$ be k-wise independent variables, and let $X := \sum_{i=1}^n X_i$, $\mu := \mathbb{E}[X]$. For any $\delta \ge 1$,*
$$\Pr\left[|X - \mu| \ge \delta\mu\right] \le \exp(-\min\{\lfloor k/2 \rfloor, \lfloor \delta\mu/3 \rfloor\}).$$

Now, we present the proof of Lemma 8.5.

Proof of Lemma 8.5. Since $L \in \mathsf{NP}_{\mathsf{sv}}$, there exist an NP-type verifier V for L, a verifier W_0 for an AM protocol, and a polynomial t such that, for every string x of length n and any $\ell \in \mathbb{N}$,

1. if $0 \le \sigma_V(x) \le \ell$, then for every r, there exists y' such that $W_0(x,\ell;y',r) = 1$, and
2. if $\sigma_V(x) > \ell + \log t(n)$, then with probability at most 2^{-3n} over the choice of $r \sim \{0,1\}^{t(n)}$, there exists $y' \in \{0,1\}^{t(n)}$ such that $W_0(x,\ell;y',r) = 1$.

We define an algorithm W as follows. Fix any string x of length n. The algorithm W takes an input (x, y) together with nondeterministic bits $y' = (y'_1, \dots, y'_{t(n)})$ and random bits $r = (r_1, \dots, r_{t(n)})$ as well as a random $4n$-wise independent hash $h \sim \mathcal{H}_{t(n),t(n),4n}$. The algorithm W computes

$$\ell_{y',r} := \min \left\{ \ell \in \{0, 1, \dots, t(n)\} \;\middle|\; W_0(x, \ell; y'_\ell, r_\ell) = 1 \right\},$$

and accepts if and only if $V(x, y) = 1$ and $h(y){\restriction}_{\ell_{y',r}-1} \in \{0\}^*$. Here, $z{\restriction}_\ell$ denotes the first ℓ bits of z for every string $z \in \{0,1\}^*$ and every integer $\ell \geq 0$; if $\ell < 0$, $h(y){\restriction}_\ell$ is defined as the empty string. Let $W(x, y; y', r, h)$ denote the output of the algorithm W.

It is easy to see Items 1 and 2 by the definitions of V and W. We prove Items 3 and 4 below.

We prepare several notations. Fix any input x and any r. Let $Y_x := \left\{ y \in \{0,1\}^{p(n)} \;\middle|\; V(x, y) = 1 \right\}$ be the set of certificates. For every $y \in Y_x$, define $I_y \in \{0, 1\}$ to be the random variable that takes 1 if and only if there exists a string y' such that $h(y){\restriction}_{\ell_{y',r}-1} \in \{0\}^*$. (Note that I_y is a random variable that is determined by the random choice of h.) Let $I := \sum_{y \in Y_x} I_y$, and let $\mu := \mathbb{E}[I]$.

We prove Item 3. Fix any input $x \in L$ of length n and random bits r. By the property of W_0, for $\ell := \sigma_V(x) \geq 0$, there exists some y'_ℓ such that $W_0(x, \ell; y'_\ell, r_\ell) = 1$, which implies that $\ell_{y',r} \leq \sigma_V(x)$ for some y'. In particular, if $h(y){\restriction}_{\sigma_V(x)-1} \in \{0\}^*$, then $h(y){\restriction}_{\ell_{y',r}-1} \in \{0\}^*$. This implies that, for every $y \in Y_x$,

$$\mathop{\mathbb{E}}_h [I_y] \geq \Pr_h \left[h(y){\restriction}_{\sigma_V(x)-1} \in \{0\}^* \right] = 2^{-\max\{0, \sigma_V(x)-1\}}.$$

We claim that $I > 0$ with probability at least $\frac{1}{2}$ by analyzing two cases: If $\sigma_V(x) = 0$, then we have $I_y = 1$ for the unique $y \in Y_x$ and thus $I > 0$. If $\sigma_V(x) \geq 1$, then $\mu \geq |Y_x| \cdot 2^{-(\sigma_V(x)-1)} = 2$. By Chebyshev's inequality, we obtain

$$\Pr_h [I = 0] \leq \Pr_h [|I - \mu| \geq \mu] \leq \frac{\mathrm{Var}[I]}{\mu^2} \leq \frac{1}{\mu} \leq \frac{1}{2},$$

and thus, with probability at least $\frac{1}{2}$ over the choice of h, we have $I > 0$, which implies that $W(x, y; y', r, h) = 1$ for some y and y'. This completes the proof of Item 3.

It remains to prove Item 4. Fix any input x of length n. Observe that, for every r and h,

$$\begin{aligned} &\# \left\{ y \in \{0,1\}^{p(n)} \;\middle|\; \exists y' \in \{0,1\}^{p(n)}, W(x, y; y', r, h) = 1 \right\} \\ &= \# \left\{ y \in Y_x \;\middle|\; \exists y' \in \{0,1\}^{p(n)}, h(y){\restriction}_{\ell_{y',r}-1} \in \{0\}^* \right\} = I. \end{aligned}$$

Thus, it suffices to upper-bound I with high probability. We may assume without loss of generality that $\sigma_V(x) \geq 0$, as otherwise $I \leq 0$. By a union bound over all $\ell \in \{0, \dots, t(n)\}$, with probability at least $1 - 2^{-3n} \cdot (t(n) + 1)$ over a random choice of r, for every $\ell < \sigma_V(x) - \log t(n)$ and every y', it holds that $W_0(x, \ell; y'_\ell, r_\ell) = 0$, in which case we have $\ell_{y',r} \geq \sigma_V(x) - \log t(n)$ for every y'. Under this event, we have

$$\mathop{\mathbb{E}}_h [I_y] = \Pr_h \left[\exists y',\ h(y){\restriction}_{\ell_{y',r}-1} \in \{0\}^* \right] \leq \Pr_h \left[h(y){\restriction}_{\sigma_V(x)-\log t(n)-1} \in \{0\}^* \right] \leq 2^{-(\sigma_V(x)-\log t(n)-1)},$$

and thus $\mu \leq |Y_x| \cdot 2^{-\sigma_V(x)+\log t(n)+1} = 2t(n)$. In this case, by Lemma 8.7, we obtain

$$\Pr_h [I \geq 4t(n) + 6n] \leq \Pr_h [I \geq 2\mu + 6n] \leq \Pr_h [|I - \mu| \geq \mu + 6n] \leq \exp(-2n).$$

Overall, choosing a polynomial $p(n)$ larger than $4t(n) + 6n$, with probability at least $1 - 2^{-2n}$ over the choice of r and h, we have $I \leq p(n)$, which completes the proof of Item 4. □

8.2 Universal Heuristic Scheme for $\mathsf{NP}_{\mathsf{sv}}$

We are now ready to present a universal heuristic scheme for $\mathsf{NP}_{\mathsf{sv}}$. In fact, we construct a universal heuristic scheme for the "search version" of $\mathsf{NP}_{\mathsf{sv}}$, which is defined as follows.

Definition 8.8. *Let V be an NP-type verifier. An algorithm A is said to solve the* search problem associated with V *if $V(x, A(x)) = 1$ for every x such that $V(x,y) = 1$ for some y. The search version of $\mathsf{NP}_{\mathsf{sv}}$ is said to* admit a universal heuristic scheme *if, for every $L \in \mathsf{NP}_{\mathsf{sv}}$ and every $\mathsf{NP}_{\mathsf{sv}}$-type verifier V for L, there exists a pair (S, C) of polynomial-time algorithms such that, for some polynomial p, for any $n \in \mathbb{N}$, any $t \geq p(n)$, and any $x \in \{0,1\}^n$,*

1. *if $\mathrm{cd}^{t,p(t)}(x) \leq k$, then $C(x, 1^t, 1^k) = 1$, and*
2. *if $C(x, 1^t, 1^k) = 1$ and $x \in L$, then $V(x, S(x, 1^t, 1^{2^k})) = 1$.*

Theorem 8.9. *If $\mathsf{coNP} \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ for some constant c, then the search version of $\mathsf{NP}_{\mathsf{sv}}$ admits a universal heuristic scheme.*

Proof. Let $L \in \mathsf{NP}_{\mathsf{sv}}$, and let V be an $\mathsf{NP}_{\mathsf{sv}}$-type verifier for L. Let W and p be the polynomial-time algorithm and the polynomial of Lemma 8.5, respectively. We define an ensemble $L' = \left\{ L'_{\langle n,k,t,s \rangle} \right\}_{n,k,t,s \in \mathbb{N}}$ of languages so that $(x, w, r) \in L'_{\langle n,k,t,s \rangle}$ if and only if there exist strings y, y', and z such that $|x| = n$, $|r| = p(n)$, $|y| = |y'| = p(n)$, $k \leq 2n$, $\mathrm{K}^t(x) \leq s$, $w = \mathrm{DP}_k(y; z)$, and $W(x, y; y', r) = 1$. It is easy to see that $L' \in \mathsf{NP}$.

We claim that the size of $L'_{\langle n,k,t,s \rangle}$ is small. Fix any $x \in \{0,1\}^n$. Let $d := p(n)k$. Pick $w \sim \{0,1\}^{d+k}$ and $r \sim \{0,1\}^{p(n)}$ randomly. We first upper-bound the probability that $(x, w, r) \in L'_{\langle n,k,t,s \rangle}$. Let R denote the set of $r \in \{0,1\}^{p(n)}$ such that the event of Item 4 of Lemma 8.5 is satisfied. Then,

$$
\begin{aligned}
&\Pr_{w,r}\left[(x, w, r) \in L'_{\langle n,k,t,s \rangle}\right] \\
&\leq \Pr_{w,r}\left[\exists y, y' \in \{0,1\}^{p(n)}, \exists z \in \{0,1\}^d, w = \mathrm{DP}_k(y; z), W(x, y; y', r) = 1\right] \\
&\leq \Pr_r[r \notin R] + \Pr_{w,r}\left[\exists y, y' \in \{0,1\}^{p(n)}, \exists z \in \{0,1\}^d, w = \mathrm{DP}_k(y; z), W(x, y; y', r) = 1 \;\middle|\; r \in R\right] \\
&\leq 2^{-2n} + \max_{r \in R} \sum_z \sum_{y \in Y_r} \Pr_w[w = \mathrm{DP}_k(y; z)],
\end{aligned}
$$

where $Y_r := \left\{ y \in \{0,1\}^{p(n)} \;\middle|\; \exists y' \in \{0,1\}^{p(n)}, W(x, y; y', r) = 1 \right\}$. By Item 4 of Lemma 8.5, we have $|Y_r| \leq p(n)$. We thus obtain

$$
\begin{aligned}
&\Pr_{w,r}\left[(x, w, r) \in L'_{\langle n,k,t,s \rangle}\right] \\
&\leq 2^{-2n} + 2^d \cdot p(n) \cdot 2^{-d-k} \leq 2^{-k+\log p(n)+1}.
\end{aligned}
$$

Summing over all strings $x \in \{0,1\}^n$ such that $\mathrm{K}^t(x) \leq s$, we have

$$
\left| L'_{\langle n,k,t,s \rangle} \right| \leq 2^{s+1} \cdot 2^{|w|+|r|} \cdot 2^{-k+\log p(n)+1},
$$

where $|w| := d + k$ and $|r| := p(n)$; thus, we obtain

$$\log\left|L'_{\langle n,k,t,s\rangle}\right| \le s + |w| + |r| - k + \log p(n) + 2.$$

By Theorem 4.2, there exist a polynomial p_Π and a polynomial-time algorithm M that solves the promise problem $\Pi = (L', \Pi_{\mathrm{No}})$, where Π_{No} consists of $(x, w, r; 1^{\langle n,k,t,s\rangle})$ such that $\mathrm{K}^{p_\Pi(t)}(x, w, r) > s + |w| + |r| - k + \log p_\Pi(t)$ and $n + k + s \le t$.[24] By Lemma 5.1, there exists a polynomial-time algorithm M_K that solves $\mathrm{Gap}_\tau\mathrm{MINKT}$ for some polynomial τ. Let $p_\mathrm{K}(n) := \tau(2n)$. Using M and M_K, we define a checker C and a solver S below.

The checker C takes $(x, 1^t, 1^k)$ as input. Let $n := |x|$. Using M_K and Fact 3.8, the checker C computes a value s such that

$$\mathrm{K}^{p_\mathrm{K}(t)}(x) \le s \le \mathrm{K}^t(x) + \log p_\mathrm{K}(t). \tag{12}$$

Let $t' := p_\mathrm{K}(t)$, and let $k' = k + O(\log t)$ be a parameter chosen later. Using the pseudorandom generator of Lemma 3.4, we compute a value v such that

$$\left| v - \Pr_{w,r}\left[M(x, w, r; 1^{\langle n,k',t',s\rangle}) = 1\right]\right| \le \epsilon, \tag{13}$$

where $\epsilon := \frac{1}{24}$. (Specifically, v is defined as $\Pr_z\left[M(x, G_m(z); 1^{\langle n,k',t',s\rangle}) = 1\right]$ for a sufficiently large m, where G_m denotes the pseudorandom generator of Lemma 3.4.) The checker C accepts if and only if $v \le 2\epsilon$.

We first describe a randomized solver S. The solver S takes $(x, 1^t, 1^{2^k})$ and random bits r as input, and computes k', s, and t' in the same way as C. We define a *deterministic* circuit D_r so that $D_r(w) := M(x, w, r; 1^{\langle n,k',t',s\rangle})$ for every w. Applying Theorem 3.12 to D_r, the solver S computes the set Y_r of all the strings y such that $\mathrm{K}^{p_\mathrm{DP}(t)}(y \mid D_r) \le k' + \log p_\mathrm{DP}(t)$, where p_DP is some polynomial. If there exists some $y \in Y_r$ such that $V(x, y) = 1$, then the solver S outputs y; otherwise, S outputs $\perp$. Note that S can be implemented as a polynomial-time algorithm, since the number of the strings $y \in Y_r$ with conditional small time-bounded Kolmogorov complexity is at most $\mathsf{poly}(n, t, 2^{k'})$, which can be enumerated in time $\mathsf{poly}(n, t, 2^k)$.

The following two claims ensure that (S, C) is a universal heuristic scheme.

Claim 8.10. *There exists a polynomial q such that, if $\mathrm{cd}^{t,q(t)}(x) \le k$, then $C(x, 1^t, 1^k) = 1$.*

Proof. Let $t'' := p_\Pi(t') = p_\Pi(p_\mathrm{K}(t))$. By Theorem 5.2, there exists a polynomial p_w such that, with probability at least $1 - \epsilon$ over the random choice of w and r, it holds that

$$\mathrm{K}^{t''}(x, w, r) \ge \mathrm{K}^{p_\mathrm{w}(t'')}(x) + |w| + |r| - \log p_\mathrm{w}(t'').$$

Choosing $q(t) := p_\mathrm{w}(t'')$, we obtain

$$\begin{aligned}
\mathrm{K}^{t''}(x, w, r) &\ge \mathrm{K}^t(x) - \left(\mathrm{K}^t(x) - \mathrm{K}^{q(t)}(x)\right) + |w| + |r| - \log p_\mathrm{w}(t'') \\
&\ge s - \log p_\mathrm{K}(t) - \mathrm{cd}^{t,q(t)}(x) + |w| + |r| - \log p_\mathrm{w}(t'') && \text{(by Eq. (12))} \\
&\ge s + |w| + |r| - k - \log p_\mathrm{K}(t) - \log p_\mathrm{w}(t'') \\
&> s + |w| + |r| - k' + \log p_\Pi(t'),
\end{aligned}$$

[24] We add this condition so that t is the largest parameter and the bound $p_\Pi(t)$ depends only on t.

where we choose $k' := k + \log p_\Pi(t') + \log p_{\mathrm{K}}(t) + \log p_{\mathrm{w}}(t'') + 1$ in the last inequality. This means that $(x, w, r; 1^{\langle n,k',t',s\rangle}) \in \Pi_{\mathrm{No}}$.

By Eq. (13), we obtain

$$v \le \Pr_{w,r}\left[M\left(x, w, r; 1^{\langle n,k',t',s\rangle}\right) = 1\right] + \epsilon \le \Pr_{w,r}\left[\left(x, w, r; 1^{n,k',t',s}\right) \notin \Pi_{\mathrm{No}}\right] + \epsilon \le 2\epsilon,$$

which implies that $C(x, 1^t, 1^k) = 1$. ◇

Claim 8.11. *If $C(x, 1^t, 1^k) = 1$ and $x \in L$, then $V(x, S(x, 1^t, 1^{2^k}; r)) = 1$ holds with probability at least $\frac{1}{4}$ over the random choice of r.*

Proof. It suffices to claim that, with probability at least $\frac{1}{4}$ over the choice of r, the deterministic circuit D_r $\frac{1}{2}$-distinguishes $\mathrm{DP}_{k'}(y_r; \text{-})$ from the uniform distribution for some certificate y_r such that $V(x, y_r) = 1$.

On the one hand, the assumption of the claim implies that $v \le 2\epsilon$; thus, by Eq. (12),

$$\Pr_{w,r}[D_r(w) = 1] \le 3\epsilon.$$

By Markov's inequality, with probability at least $\frac{3}{4}$ over r,

$$\Pr_{w}[D_r(w) = 1] \le 12\epsilon \le \frac{1}{2}.$$

On the other hand, by Items 2 and 3 of Lemma 8.5, with probability at least $\frac{1}{2}$ over the random choice of r, there exist strings y_r, y_r' such that $W(x, y_r; y_r', r) = 1$ and $V(x, y_r) = 1$. Under this event, we have $(x, \mathrm{DP}_{k'}(y_r; z), r) \in L'_{\langle n,k',t',s\rangle}$ for every z since $\mathrm{K}^{t'}(x) \le s$ by Eq. (12); thus, we also have $D_r(\mathrm{DP}_{k'}(y_r; z)) = 1$.

By a union bound, with probability at least $1 - \frac{1}{4} - \frac{1}{2}$ over the choice of r, there exists y_r such that $V(x, y_r) = 1$ and

$$\Pr_{z}[D_r(\mathrm{DP}_{k'}(y_r; z)) = 1] - \Pr_{w}[D_r(w) = 1] \ge \frac{1}{2}.$$

◇

Finally, the solver S can be derandomized using Lemma 3.4: Let S' be an algorithm that takes $(x, 1^t, 1^{2^k})$ as input and outputs $y = S(x, 1^t, 1^{2^k}; G_m(\sigma))$ such that $V(x, y) = 1$ (if any) using an exhaustive search over all seeds $\sigma \in \{0,1\}^{O(\log m)}$, where G_m is the pseudorandom generator of Lemma 3.4 and $m = \mathsf{poly}(n, t, 2^k)$. Then, S' is a polynomial-time algorithm such that $C(x, 1^t, 1^k) = 1$ implies $V(x, S'(x, 1^t, 1^{2^k})) = 1$ for every $x \in L$. (Otherwise, there exists a small circuit A defined as $A(r) := V(x, S(x, 1^t, 1^{2^k}; r))$ that distinguishes $G_m(\text{-})$ from the uniform distribution.) We conclude that (S, C') is a (deterministic) universal heuristic scheme for the search problem. □

Since a decision problem reduces to its search version, we obtain the following corollary:

Corollary 8.12. *If $\mathsf{coNP} \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ for some constant c, then every $L \in \mathsf{NP}_{\mathsf{sv}}$ admits a universal heuristic scheme.*

8.3 Approximately Size-Verifiable One-Way Function

In this subsection, we relate the search version of $\mathsf{NP}_{\mathsf{sv}}$ to the task of inverting an approximately size-verifiable one-way function in the worst case. The definition of inverting a function in the worst case is given below.

Definition 8.13. *Let $f = \left\{f_n : \{0,1\}^{m(n)} \to \{0,1\}^n\right\}_{n\in\mathbb{N}}$ be a family of functions. An algorithm A is said to* invert f (in the worst case) *if $A(f_n(x)) \in f_n^{-1}(f_n(x))$ for every $n \in \mathbb{N}$ and every $x \in \{0,1\}^{m(n)}$. If there is no polynomial-time algorithm that inverts f, then f is said to be* (worst-case) one-way*.*

We now define an approximately size-verifiable one-way function.

Definition 8.14. *For a function $t\colon \mathbb{N} \to \mathbb{N}$, a family $f = \left\{f_n : \{0,1\}^{t(n)} \to \{0,1\}^n\right\}_{n\in\mathbb{N}}$ of functions is said to be $t(n)$-*time approximately size-verifiable *if the following conditions hold.*

1. *There exists an algorithm that takes $n \in \mathbb{N}$ and $y \in \{0,1\}^{t(n)}$ as input and outputs $f_n(y)$ in time $O(t(n))$.*
2. *The promise problem Π^V defined in Definition 8.2 is in $\mathrm{pr}\text{-}\mathsf{AMTIME}(t(n))$, where V is the algorithm that takes $x \in \{0,1\}^*$ and $y \in \{0,1\}^{t(|x|)}$ as input and accepts if and only if $f_{|x|}(y) = x$.*

We observe that inverting an approximately size-verifiable one-way function reduces to solving the search version of $\mathsf{NP}_{\mathsf{sv}}$.

Proposition 8.15. *For every polynomial-time approximately size-verifiable function f, there exists an $\mathsf{NP}_{\mathsf{sv}}$-type verifier V such that inverting f in the worst case is polynomial-time-reducible to solving the search problem associated with V.*[25]

Proof Sketch. We define V so that $V(x,y) := 1$ if and only if $f_{|x|}(y) = x$. It is immediate that V is an $\mathsf{NP}_{\mathsf{sv}}$-type verifier. Suppose that an algorithm A solves the search problem associated with V. Fix any $n \in \mathbb{N}$ and consider any y in the domain of f_n. Let $x := f_n(y)$. We have $V(x,y) = 1$; thus, $A(x)$ finds y' such that $V(x,y') = 1$, which implies that $f_n(y') = x$. It follows that $A(f_n(y)) = A(x) = y' \in f_n^{-1}(f_n(y))$. □

Theorem 8.9 and Proposition 8.15 imply that there exists a universal heuristic scheme for inverting any polynomial-time approximately size-verifiable function under the assumption that NP is easy on average.

9 P-Computable Average-Case Polynomial Time

In this section, we study the notion of P-computable average-case polynomial time.

Definition 9.1 ($\mathsf{Avg_P P}$; P-Computable Average-Case Polynomial Time)**.** *We say that an algorithm A has* P-computable average-case polynomial running time *with respect to a family $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ of distributions if there exist a polynomial-time computable function $t\colon \{0,1\}^* \to \mathbb{N}$ and a constant $\epsilon > 0$ such that, for every $n \in \mathbb{N}$,*

[25] We mention that a partial converse of Proposition 8.15 holds: For every $\mathsf{NP}_{\mathsf{sv}}$-type verifier V, there exists a polynomial-time approximately size-verifiable function f computable with advice strings of $O(\log n)$ bits such that the search version associated with V is reducible to inverting f in the worst case.

1. $\mathbb{E}_{x\sim\mathcal{D}_n}[t(x,1^n)^\epsilon] \le O(n)$ and

2. $A(x,1^n)$ halts in time $t(x,1^n)$ for every $x \in \mathrm{supp}(\mathcal{D}_n)$.

Let $\mathsf{Avg_P P}$ denote the class of distributional problems $(L,\mathcal{D})$ such that there exists an algorithm A such that $A(x,1^n) = L(x)$ for every $n \in \mathbb{N}$ and every $x \in \mathrm{supp}(\mathcal{D}_n)$, and A has P-computable average-case polynomial running time.

Using Markov's inequality, it is easy to see the following equivalent conditions of Item 1.

Fact 9.2 (cf. [BT06a])**.** *For any function $t\colon \{0,1\}^* \to \mathbb{N}$ and a family $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ of distributions, the following are equivalent:*

1. There exists a constant $\epsilon > 0$ such that $\mathbb{E}_{x\sim\mathcal{D}_n}[t(x,1^n)^\epsilon] \le O(n)$ for every $n \in \mathbb{N}$.

2. There exists a constant $\epsilon > 0$ such that $\mathbb{E}_{x\sim\mathcal{D}_n}[t(x,1^n)^\epsilon] \le n^{O(1)}$ for every $n \in \mathbb{N}$.

3. There exist a constant $\epsilon > 0$ and a polynomial p such that

$$\Pr_{x\sim\mathcal{D}_n}[t(x,1^n) \ge t] \le \frac{p(n)}{t^\epsilon}$$

for every $n \in \mathbb{N}$ and every $t \in \mathbb{N}$.

9.1 P-Bounded Failure Heuristic Scheme

We introduce the notion of P-bounded failure heuristic scheme.

Definition 9.3 (P-Bounded Failure Heuristic Scheme)**.** *A* P-bounded failure heuristic scheme *for a distributional problem $(L,\mathcal{D})$ is a pair (S,C) of polynomial-time algorithms (referred to as a* solver *and a* checker*, respectively) such that, for every n and $k \in \mathbb{N}$,*

1. for every $x \in \mathrm{supp}(\mathcal{D}_n)$, if $C(x,1^n,1^k) = 1$, then $S(x,1^n,1^{2^k}) = L(x)$, and

2. $\Pr_{x\sim\mathcal{D}_n}\left[C(x,1^n,1^k) = 1\right] \ge 1 - 2^{-k}$.

We show the equivalence between a P-computable average-case polynomial-time algorithm and a P-bounded failure heuristic scheme.

Proposition 9.4. *The following are equivalent for every language L and every polynomial-time samplable family $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ of distributions.*

1. There exists a P-bounded failure heuristic scheme for $(L,\mathcal{D})$.

2. $(L,\mathcal{D}) \in \mathsf{Avg_P P}$.

Proof. (Item 1 $\Rightarrow$ 2) Assume that there exist an algorithm A and a polynomial-time computable function $t\colon \{0,1\}^* \to \mathbb{N}$ that satisfy Definition 9.1. We define a P-bounded failure heuristic scheme (S,C) as follows. Fix any $n \in \mathbb{N}$, $x \in \mathrm{supp}(\mathcal{D}_n)$, and $k \in \mathbb{N}$. Let $s \le \mathsf{poly}(n,2^k)$ be some parameter chosen later. The checker C accepts an input $(x,1^n,1^k)$ if and only if $t(x,1^n) \le s$. The solver S accepts an input $(x,1^n,1^{2^k})$ if and only if A halts and outputs 1 on input $(x,1^n)$ in time s.

We claim that (S,C) is a P-bounded failure heuristic scheme for $(L,\mathcal{D})$.

1. Since $s \le \mathsf{poly}(n, 2^k)$, S and C are polynomial-time algorithms.

2. If $C(x, 1^n, 1^k) = 1$, then $t(x, 1^n) \le s$; thus, A halts in time s, in which case $S(x, 1^n, 1^k) = A(x, 1^n) = L(x)$.

3. By the property of (A, t), there exist a constant $\epsilon > 0$ and a polynomial p such that

$$\Pr_{x \sim \mathcal{D}_n}\left[C(x, 1^n, 1^k) = 0\right] = \Pr_{x \sim \mathcal{D}_n}\left[t(x, 1^n) > s\right] \le \frac{p(n)}{s^\epsilon} = 2^{-k},$$

where, in the last inequality, we choose $s := 2^{(k + \log p(n))/\epsilon}$.

(Item 2 $\Rightarrow$ 1) Let (S, C) be a P-bounded failure heuristic scheme for $(L, \mathcal{D})$. Since $\mathcal{D}$ is polynomial-time samplable, there exists a polynomial p such that, for every $n \in \mathbb{N}$ and every $x \in \operatorname{supp}(\mathcal{D}_n)$, $\mathcal{D}_n(x) > 2^{-p(n)}$.

We define an algorithm A as follows. On input $(x, 1^n)$, the algorithm A finds the minimum integer $k \in [p(n)]$ such that $C(x, 1^n, 1^k) = 1$. Then A outputs $S(x, 1^n, 1^{2^k})$ and halts. Clearly, $A(x, 1^n)$ outputs $L(x)$ correctly for every $n \in \mathbb{N}$ and every $x \in \operatorname{supp}(\mathcal{D}_n)$.

We first claim that A is well defined in the sense that there exists an integer $k \in [p(n)]$ such that $C(x, 1^n, 1^k) = 1$. Fix any $n \in \mathbb{N}$ and $x \in \mathcal{D}_n$ and let $k := p(n)$. Assume, toward a contradiction, that $C(x, 1^n, 1^k) = 0$. Then we have $2^{-p(n)} < \Pr_{x \sim \mathcal{D}_n}\left[C(x, 1^n, 1^k) = 0\right] \le 2^{-k}$, which is a contradiction. We conclude that $C(x, 1^n, 1^{p(n)}) = 1$.

Next, we define a function $t \colon \{0,1\}^* \to \mathbb{N}$. Since S and C are polynomial-time algorithms, there exists a polynomial such that, for every $n \in \mathbb{N}$ and every $x \in \operatorname{supp}(\mathcal{D}_n)$, A halts on input $(x, 1^n)$ in time $p(n, 2^{k^*})$, where $k^* = \min\left\{\, k \in [p(n)] \;\middle|\; C(x, 1^n, 1^k) = 1 \,\right\}$. We define $t(x, 1^n)$ to be $p(n, 2^{k^*})$. By the definition, it is clear that $A(x, 1^n)$ halts in time $t(x, 1^n)$ for every $n \in \mathbb{N}$ and $x \in \operatorname{supp}(\mathcal{D}_n)$. Observe that t is computable in polynomial time.

We claim that $\mathbb{E}\left[t(x, 1^n)^\epsilon\right] \le n^{O(1)}$ for some constant $\epsilon > 0$. Take a constant c such that $p(n, 2^k) \le 2^{ck + c\log n}$ for all large $k, n \in \mathbb{N}$, and let $\epsilon := 1/c$. Then, for any $n \in \mathbb{N}$,

$$\begin{aligned}
\mathop{\mathbb{E}}_{x \sim \mathcal{D}_n}\left[t(x, 1^n)^\epsilon\right] &\le \sum_{k=1}^{p(n)} p(n, 2^k)^\epsilon \Pr_x\left[C(x, 1^n, 1^{k-1}) = 0\right] \\
&\le \sum_{k=1}^{p(n)} \left(2^{ck + c\log n}\right)^\epsilon 2^{-(k-1)} + O(1) \le O(p(n)n).
\end{aligned}$$

□

9.2 Universality of $\mathsf{Avg_P P}$-Universal Heuristic Schemes

We justify the name of "universal" heuristic scheme by presenting the following characterization of a universal heuristic scheme.

Theorem 9.5 (Universality of $\mathsf{Avg_P P}$-Universal Heuristic Scheme)**.** *Assume that* $\mathsf{coNP} \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ *for some constant c. Then, the following are equivalent for any language L:*

1. *There exists a universal heuristic scheme for L.*

2. $\{L\} \times \mathrm{PS}\small{\mathrm{AMP}} \subseteq \mathsf{Avg_P P}$.

The implication from Item 1 to Item 2 is provided below.

Theorem 9.6. *Assume that* $\mathsf{coNP} \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ *for some constant* c*. If there exists a universal heuristic scheme for a language* L*, then* $\{L\} \times \mathrm{PSAMP} \subseteq \mathsf{Avg}_\mathsf{P}\mathsf{P}$*.*

Lemma 9.7 ([AF09, AGvM$^+$18])**.** *Let* $\mathcal{D}$ *be any polynomial-time samplable family of distributions. Then, there exist polynomials* p *and* q *such that, for every* $n, s \in \mathbb{N}$ *and every* $x \in \mathrm{supp}(\mathcal{D}_n)$ *with* $\mathcal{D}_n(x) \geq 2^{-s}$*,*

$$\Pr_{r \sim \{0,1\}^{q(n)}}\left[\mathrm{K}^{p(n)}(x, r) \leq s + |r| + \log p(n)\right] \geq \frac{1}{4}.$$

There are at least two proofs of Lemma 9.7. Antunes and Fortnow [AF09] used the Nisan–Wigderson generator for AC^0 [NW94]. We present a proof based on a pairwise-independent hash family [AGvM$^+$18].

Proof of Lemma 9.7. Let R be a polynomial-time algorithm such that, for every $n \in \mathbb{N}$, the distribution of $R(1^n, r)$ for $r \sim \{0,1\}^{p(n)}$ is identical to $\mathcal{D}_n$, where p is some polynomial. Let $\ell := p(n)$ and $m := \ell - s - 2$.

We define a pairwise-independent hash family $\mathcal{H} = \left\{h \colon \{0,1\}^\ell \to \{0,1\}^m\right\}$ so that $h(\sigma) := U \cdot \sigma + v$, where $U \sim \mathrm{GF}(2)^{m \times \ell}$ is a binary matrix and $v \sim \mathrm{GF}(2)^m$ is a binary vector.

Let $S := \left\{ \sigma \in \{0,1\}^{p(n)} \;\middle|\; R(1^n, \sigma) = x \right\}$. Using Chebyshev's inequality, with probability at least $\frac{1}{4}$, we have $1 \leq \left|S \cap h^{-1}(0^m)\right|$ and $\left|h^{-1}(0^m)\right| \leq 2^{s+3}$ (cf. [AGvM$^+$18, Claim 4.2.1]). Using Gaussian elimination, every $x \in h^{-1}(0^m)$ can be described by h and $\log\left|h^{-1}(0^m)\right| \leq s + 3$ bits of information (cf. [AGvM$^+$18, Claim 4.2.2]). In particular, by choosing $\sigma \in S \cap h^{-1}(0^m)$, x can be described by (U, v) and $s + 3$ bits; thus, $\mathrm{K}^{\mathsf{poly}(n)}(x, U, v) \leq s + 3 + m\ell + m + O(\log n)$. □

Corollary 9.8. *Assume that* $\mathsf{coNP} \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ *for some constant* c*. Let* $\mathcal{D}$ *be any polynomial-time samplable family of distributions. Then there exists a polynomial* p *such that, for every* $n, s \in \mathbb{N}$ *and every* $x \in \mathrm{supp}(\mathcal{D}_n)$*, if* $\mathcal{D}_n(x) \geq 2^{-s}$*, then* $\mathrm{K}^{p(n)}(x) \leq s + \log p(n)$*.*

Proof. By Lemma 9.7, with probability at least $\frac{1}{4}$ over a random choice of r, $\mathrm{K}^{p(n)}(x, r) \leq s + |r| + \log p(n)$. By Theorem 5.2, there exists a polynomial p_{w} such that with probability at least $\frac{7}{8}$ over a random choice of r, $\mathrm{K}^{p(n)}(x, r) \geq \mathrm{K}^{p_{\mathrm{w}}(p(n))}(x) + |r| - \log p_{\mathrm{w}}(p(n))$. By a union bound, there exists r that satisfies both inequalities, from which it follows that $\mathrm{K}^{p_{\mathrm{w}}(p(n))}(x) \leq s + \log p(n) + \log p_{\mathrm{w}}(p(n))$. □

Proof of Theorem 9.6. Take any $\mathcal{D} \in \mathrm{PSAMP}$. Fix any $n \in \mathbb{N}$. We claim that $\Pr_{x \sim \mathcal{D}_n}\left[\mathrm{cd}^{p(n)}(x) > k\right]$ is exponentially small in k for some polynomial p. For every $s \in \mathbb{N}$, let

$$A_s := \left\{ x \in \mathrm{supp}(\mathcal{D}_n) \;\middle|\; 2^{-s-1} < \mathcal{D}(x) \leq 2^{-s} \right\}.$$

Since $\mathcal{D} \in \mathrm{PSAMP}$, there exists a polynomial q such that $\mathcal{D}(x) \geq 2^{-q(n)}$ holds for every $x \in \mathcal{D}_n$. Thus, $\mathrm{supp}(\mathcal{D}_n) = \bigcup_{s \leq q(n)} A_s$.

By Corollary 9.8, there exists a polynomial p such that $\mathrm{K}^{p(n)}(x) \leq s + 1 + \log p(n)$ for every $x \in A_s$. Let

$$B := \left\{ x \in \mathrm{supp}(\mathcal{D}_n) \;\middle|\; \mathrm{cd}^{p(n)}(x) > k \right\}.$$

Consider any $x \in A_s \cap B$. Since $\mathrm{K}^{p(n)}(x) - \mathrm{K}(x) > k$, we obtain $\mathrm{K}(x) < s - k + 1 + \log p(n)$. By Fact 3.7, $|A_s \cap B| \leq 2^{s-k+1+\log p(n)}$. Therefore,

$$\Pr_{x \in \mathcal{D}_n}[x \in A_s \cap B] \leq 2^{-s} \cdot |A_s \cap B|. \leq 2^{-k+1+\log p(n)}.$$

By taking a union bound over all $s \leq q(n)$,

$$\Pr_{x \in \mathcal{D}_n}[x \in B] \leq q(n) \cdot 2^{-k+1+\log p(n)}.$$

Let $\kappa(k,n) := k + 1 + \log p(n)q(n)$.

Let (S, C) be a universal heuristic scheme for L. We define a pair (S', C') of polynomial-time algorithms so that, for every x of length n and every k,

$$C'(x, 1^n, 1^k) := C(x, 1^{p(n)}, 1^{\kappa(k,n)}),$$
$$S'(x, 1^n, 1^{2^k}) := S(x, 1^{p(n)}, 1^{2^{\kappa(k,n)}}).$$

We claim that (S', C') is a P-bounded failure average-case heuristic scheme for $(L, \mathcal{D})$. The correctness of S' is obvious. Since $\mathrm{cd}^{p(n),\mathsf{poly}(p(n))} \leq \mathrm{cd}^{p(n)}(x)$, we also have

$$\Pr_{x \sim \mathcal{D}_n}\left[C'(x, 1^n, 1^k) = 0\right] \leq \Pr_{x \sim \mathcal{D}_n}\left[\mathrm{cd}^{p(n),\mathsf{poly}(p(n))}(x) > k\right] \leq 2^{-\kappa(k,n)+1+\log p(n)q(n)} = 2^{-k}.$$

□

Next, we prove the converse. The following theorem establishes the implication from Item 2 to Item 1 of Theorem 9.5.

Theorem 9.9. *There exists a polynomial-time samplable family $\mathcal{M}$ of distributions such that, for every language L, if $(L, \mathcal{M}) \in \mathsf{Avg_P P}$ and $\mathsf{coNP} \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ for some constant c, then L admits a universal heuristic scheme.*

The following corollary is an immediate consequence of Theorem 9.9.

Corollary 9.10. *If $\mathsf{DistNP} \subseteq \mathsf{Avg_P P}$, then every $L \in \mathsf{NP}$ admits a universal heuristic scheme.*

Proof. Observe that $\mathsf{DistNP} \subseteq \mathsf{Avg_P P}$ implies $(L, \mathcal{M}) \in \mathsf{Avg_P P}$ for every $L \in \mathsf{NP}$ and $\mathsf{Dist(coNP)} \subseteq \mathsf{Avg}^1\mathsf{P}$. □

Proof of Theorem 9.9. The idea of the proof is as follows. We take a polynomial-time samplable family $\mathcal{M}$ that "dominates the time-bounded universal distribution". We then consider a P-bounded failure heuristic scheme (S, C) with respect to $\mathcal{M}$. Since the number of inputs on which S fails is small, using the language compression theorem, we conclude that any input on which S fails has small time-bounded Kolmogorov complexity. Details follow.

We define a polynomial-time samplable family $\mathcal{M} = \{\mathcal{M}_{\langle n,t \rangle}\}_{n,t \in \mathbb{N}}$ of distributions by the following randomized algorithm: On input $1^{\langle n,t \rangle}$, let $n' := 2^{\lceil \log n \rceil}$.[26] Pick $s \sim [2n']$ and $d \sim \{0,1\}^s$ randomly, and simulate the universal Turing machine U on input d. If U halts on input d in time t and outputs a string x of length n, then output x; otherwise, output 1^n. Observe that $\mathcal{M}_{\langle n,t \rangle}(x) \geq \frac{1}{4n} \cdot 2^{-\mathrm{K}^t(x)}$ because x is sampled when $s = \mathrm{K}^t(x)$ (which happens with probability $\geq \frac{1}{4n}$) and d is a description of x (which happens with probability $\geq 2^{-s}$).

Let (S, C) be a P-bounded failure heuristic scheme for $(L, \mathcal{M})$.

[26] We make n' a power of 2 in order to ensure that $\mathcal{M}$ is polynomial-time samplable according to Definition 3.1.

Claim 9.11. *There exists a polynomial p such that, for every $n \in \mathbb{N}$, every $x \in \{0,1\}^n$, every $t \geq n$ and $k \leq t$, $C(x, 1^{\langle n,t \rangle}, 1^k) = 0$ implies $\mathrm{cd}^{t,p(t)}(x) > k - \log p(t)$.*

Proof. Using Lemma 5.1, let M_{K} be a polynomial-time algorithm for $\mathrm{Gap}_\tau\mathrm{MINKT}$, where τ is some polynomial. Define F to be an ensemble of languages such that

$$F_{\langle n,t,k,s \rangle} := \left\{ x \in \{0,1\}^n \;\middle|\; M_{\mathrm{K}}(x, 1^t, 1^s) = 1 \text{ and } C(x, 1^{\langle n, \tau(n+t) \rangle}, 1^k) = 0 \right\}.$$

Fix any $n, t, k, s \in \mathbb{N}$, and let $t' := \tau(n+t)$. By the property of the checker C,

$$\begin{aligned} 2^{-k} &\geq \Pr_{x \sim \mathcal{M}_{\langle n,t' \rangle}} \left[C(x, 1^{\langle n,t' \rangle}, 1^k) = 0 \right] \\ &\geq \sum_{x \in F_{\langle n,t,k,s \rangle}} \mathcal{M}_{\langle n,t' \rangle}(x) \\ &\geq \sum_{x \in F_{\langle n,t,k,s \rangle}} \frac{1}{4n} \cdot 2^{-\mathrm{K}^{t'}(x)} \\ &\geq \sum_{x \in F_{\langle n,t,k,s \rangle}} 2^{-s - \log(4nt')}, \end{aligned}$$

where, in the last inequality, we used the fact that $\mathrm{K}^{\tau(n+t)}(x) \leq s + \log \tau(n+t)$ for every $x \in F_{\langle n,t,k,s \rangle}$. Thus, we obtain $|F_{\langle n,t,k,s \rangle}| \leq 2^{s-k+\log(4nt')}$. By applying Corollary 4.4 to $F_{\langle n,t,k,s \rangle}$, there exists a polynomial q_F such that $\mathrm{K}^{q_F(n+t+k+s)}(x) \leq s - k + \log(4nt') + \log q_F(n+t+k+s)$ for every $x \in F_{\langle n,t,k,s \rangle}$. Since t is the largest parameter among n, t, and s, there exists a polynomial p such that, for every $x \in F_{\langle n,t,k,s \rangle}$,

$$\mathrm{K}^{p(t)}(x) < s - k + \log p(t).$$

Now, fix any string x of length n, and let $s := \mathrm{K}^t(x) \leq O(n)$. Since $M_{\mathrm{K}}(x, 1^t, 1^s) = 1$, we have $x \in F_{\langle n,t,k,s \rangle}$; thus, $\mathrm{K}^{p(t)}(x) < \mathrm{K}^t(x) - k + \log p(t)$, which is equivalent to $k - \log p(t) < \mathrm{cd}^{t,p(t)}(x)$. $\diamond$

Now we define a pair (S', C') of polynomial-time algorithms as follows.

$$\begin{aligned} C'(x, 1^t, 1^k) &:= C(x, 1^{\langle |x|, t \rangle}, 1^{k + \log p(t)}), \\ S'(x, 1^t, 1^k) &:= S(x, 1^{\langle |x|, t \rangle}, 1^{2^{k + \log p(t)}}). \end{aligned}$$

By Claim 9.11, $\mathrm{cd}^{t,p(t)}(x) \leq k$ implies $C'(x, 1^t, 1^k) = 1$. It is easy to see that (S', C') is a universal heuristic scheme.[27] $\square$

10 Uniform Hard Distribution for $\mathsf{Avg_P P}$

In this section, we present the result that improves Corollary 9.10.

Theorem 10.1. *If $\mathsf{NP} \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{Avg_P P}$, then every language $L \in \mathsf{NP}$ admits a universal heuristic scheme.*

[27] Note that we may assume without loss of generality that $k \leq O(n)$, since $\mathrm{cd}^{t,p(t)}(x) \leq O(n)$ for a sufficiently large t.

The proof is based on an efficient compression algorithm: We show that if NP is easy on average, then there is an efficient algorithm that compresses a string nearly optimally. (This corresponds to a search version of GapMINKT.)

Theorem 10.2. *Assume that* $\mathsf{coNP} \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{Avg}^{1}_{1-n^{-c}}\mathsf{P}$ *for some constant c. Then, there exist polynomial-time algorithms E and U and a polynomial p such that*

1. *for every $t \in \mathbb{N}$ and for every string $x \in \{0,1\}^*$ with $p(|x|) \le t$, it holds that $U(E(x, 1^t), 1^t) = x$, and*

2. $|E(x, 1^t)| \le \mathrm{K}^t(x) + \log p(t)$.

Proof. We define the polynomial-time algorithm U so that $U(d, 1^t)$ is equal to the output of a universal Turing machine on input d if it halts in time $\mathsf{poly}(t)$, where poly is some large polynomial.

Using Lemma 5.1, let M_{K} be a polynomial-time algorithm that solves $\mathrm{Gap}_\tau\mathrm{MINKT}$ for some polynomial τ.

We define the polynomial-time algorithm E as follows. Fix any input $(x, 1^t)$, and let $n := |x|$. Let $\epsilon := 1/2$, and let t' be a parameter chosen later. For each $k \in [2n]$ in increasing order, E operates as follows. Define an oracle D so that $D(w) = 1$ if and only if $M_{\mathrm{K}}(w, 1^{2t}, 1^{\theta}) = 1$, where $w \in \{0,1\}^{nk+k}$ and $\theta := nk + k - \log\tau(4t) - 2$. Note that D can be computed using a circuit of size at most s, where $s \le \mathsf{poly}(t)$. Run the oracle procedure C^D of Theorem 3.12 on input $(x, 1^k, 1^{\epsilon^{-1}}, 1^s)$, and let M denote the output (which means that (M, D) is a candidate description for x). If $U((M, D), 1^{t'}) = x$, the algorithm E outputs the following program M_0 and halts: The program M_0 takes (n, k, t, θ) as input, computes the circuit D, and outputs $U((M, D), 1^{t'})$. Observe that $|M_0| \le |M| + O(\log t)$ and $U(M_0, 1^{t''}) = x$ for some large t''.

We claim the correctness of the algorithm while choosing the parameters. We assume that $t \ge p_0(n)$ for some large polynomial p_0. Observe that, for every string z,

$$\mathrm{K}^{2t}(\mathrm{DP}_k(x; z)) \le \mathrm{K}^t(x) + d + O(\log n) \le \theta,$$

where $d := |z| = nk$ and the last inequality holds for every k such that $\mathrm{K}^t(x) + O(\log n) + \log\tau(4t) \le k$; we choose the minimum integer $k = \mathrm{K}^t(x) + O(\log n) + \log\tau(4t)$ that satisfies this inequality. This implies that $D(\mathrm{DP}_k(x; z)) = 1$ for every z. On the other hand, we have $\Pr_w[D(w) = 0] \ge \frac{1}{2}$ because any input $(w, 1^{2t}, 1^{\theta})$ with $\mathrm{K}(w) > \theta + \log\tau(4t)$ is a NO instance of $\mathrm{Gap}_\tau\mathrm{MINKT}$. These properties of D indicate that D $\frac{1}{2}$-distinguishes the output distribution of $\mathrm{DP}_k(x; \text{-})$ from the uniform distribution, from which we conclude that the procedure $C^D(x, 1^k, 1^{\epsilon^{-1}}, 1^s)$ outputs some program M such that $|M| \le k + \log t' = \mathrm{K}^t(x) + O(\log t)$ and $U((M, D), 1^{t'}) = x$, where $t' \le \mathsf{poly}(t)$. $\square$

Proof of Theorem 10.1. Let (E, U) be the pair of polynomial-time algorithms given in Theorem 10.2. We define a language L' so that $\alpha \in L'$ if and only if $\langle s, t\rangle := |\alpha|$ and $U(\beta, 1^t) \in L$, where β is the first s bits of α. Observe that $L' \in \mathsf{NP}$. Let (S', C') be a P-bounded failure heuristic scheme for $(L', \mathcal{U}) \in \mathsf{Avg_P P}$.

We define a universal heuristic scheme (S, C) for L. The checker C takes $(x, 1^t, 1^k)$ as input and accepts if and only if there exists a seed $z \in \{0,1\}^{O(\log m)}$ such that $C'(E(x, 1^t)G_m(z), 1^{k'}) = 1$,[28] where m and k' are parameters chosen later, and G_m is the pseudorandom generator of Lemma 3.4

[28] More precisely, $C'(E(x, 1^t)G_m(z), 1^{|E(x,1^t)G_m(z)|}, 1^{k'}) = 1$; however, the second argument is clearly redundant.

whose output is truncated to a string of length $\langle |E(x, 1^t)|, t\rangle - |E(x, 1^t)|$. The solver S takes $(x, 1^t, 1^{2^k})$ as input, and outputs $S'(E(x, 1^t)G_m(z), 1^{2^{k'}})$, where z is the seed found by the checker.

We claim the correctness of S. Assume that $C(x, 1^t, 1^k) = 1$. Then, for some z, we have

$$C'(E(x, 1^t)G_m(z), 1^{k'}) = 1,$$

which implies that

$$\begin{aligned} S(x, 1^t, 1^{2^k}) &= S'(E(x, 1^t)G_m(z), 1^{2^{k'}}) \\ &= L'(E(x, 1^t)G_m(z)) \\ &= L(U(E(x, 1^t), 1^t)) = L(x). \end{aligned}$$

This establishes the correctness of the solver S.

It remains to prove the correctness of the checker C. By the property of (S', C'), for every $s, t \in \mathbb{N}$,

$$\Pr_{d,r}\left[C'(dr, 1^{k'}) = 0\right] \leq 2^{-k'},$$

where $d \sim \{0,1\}^s$ and $r \sim \{0,1\}^{\langle s,t\rangle - s}$. In particular, the number of strings $dr \in \{0,1\}^{\langle s,t\rangle}$ such that $C'(dr, 1^{k'}) = 0$ is at most $2^{\langle s,t\rangle - k'}$. By Corollary 4.4, there exists a polynomial p such that, for every string $dr \in \{0,1\}^{\langle s,t\rangle}$ with $C'(dr, 1^{k'}) = 0$, it holds that

$$\mathrm{K}^{p(t)}(dr) \leq \langle s, t\rangle - k' + \log p(t). \tag{14}$$

Now fix any string x and r such that $C'(E(x, 1^t)r, 1^{k'}) = 0$, and let $d := E(x, 1^t)$ and $s := |d|$. By Eq. (14), we have $\mathrm{K}^{p(t)}(dr) \leq |r| + s - k' + \log p(t)$, where $|r| = \langle s, t\rangle - s$. By Theorem 10.2, we also have $s \leq \mathrm{K}^t(x) + \log p(t)$ by choosing a sufficiently large p. Combining these two inequalities, we obtain

$$\mathrm{K}^{p(t)}(dr) \leq \mathrm{K}^t(x) + |r| - k' + 2\log p(t).$$

By Theorem 5.2, for some polynomial p_{w}, with high probability over a random choice of $r \sim \{0,1\}^{\langle s,t\rangle - s}$, it holds that

$$\mathrm{K}^{p(t)}(dr) \geq \mathrm{K}^{p_{\mathrm{w}}(p(t))}(d) + |r| - \log p_{\mathrm{w}}(t).$$

Since $U(d, 1^t) = x$, we also have $\mathrm{K}^{q(t)}(x) \leq \mathrm{K}^{p_{\mathrm{w}}(p(t))}(d) + \log p(t)$ for some polynomial q. Combining the three inequalities above, we obtain

$$k' \leq \mathrm{cd}^{t,q(t)}(x) + \log p_{\mathrm{w}}(t) + 3\log p(t).$$

By choosing $k' := k + \log p_{\mathrm{w}}(t) + 3\log p(t) + 1$, the last inequality is equivalent to $k + 1 \leq \mathrm{cd}^{t,q(t)}(x)$.

Now, assume that $\mathrm{cd}^{t,q(t)}(x) \leq k$. By the contrapositive of the argument above,

$$\Pr_{r \sim \{0,1\}^{\langle s,t\rangle - s}}\left[C'(E(x, 1^t)r, 1^{k'}) = 1\right] \approx 1.$$

By the security of the pseudorandom generator G_m (and choosing a sufficiently large m), there exists a seed z such that

$$C'(E(x, 1^t)G_m(z), 1^{k'}) = 1,$$

which implies that $C(x, 1^t, 1^k) = 1$. This establishes the correctness of the checker C. □

As a corollary, we obtain an analogue of Impagliazzo and Levin's results [IL90] in the setting of $\mathsf{Avg_P P}$.

Corollary 10.3. $\mathsf{DistNP} \subseteq \mathsf{Avg_P P} \iff \mathsf{NP} \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{Avg_P P}$.

Proof. This follows from Theorems 9.6 and 10.1. □

11 Proofs of Main Results

We combine the results presented thus far and prove the main theorems stated in Section 1. We start with a "decision version" of Item 1 of Theorem 1.8.

Theorem 11.1. *For every time-constructible function $t\colon \mathbb{N} \to \mathbb{N}$ with $n \le t(n) < 2^n$ and for every constant c,*

$$\mathsf{BP} \cdot \mathsf{NTIME_{sv}}(t(n)) \not\subseteq \mathsf{DTIME}\left(2^{O(n/\log(n/\log t(n)))}\right) \quad \Longrightarrow \quad \mathsf{coNP} \times \{\mathcal{U}, \mathcal{T}\} \not\subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}.$$

Proof. We prove the contrapositive. Assume that $\mathsf{coNP} \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$. By Corollary 8.12, every language in $\mathsf{NP_{sv}}$ admits a universal heuristic scheme. By Lemma 7.10, every language in $\mathsf{BP} \cdot \mathsf{NP_{sv}}$ admits a universal heuristic scheme. By Corollary 6.5, we obtain

$$\mathsf{BP} \cdot \mathsf{NTIME_{sv}}(t(n)) \subseteq \mathsf{DTIME}\left(2^{O(n/\log(n/\log t(n)))}\right).$$

□

Since $\mathsf{UP} \subseteq \mathsf{NP_{sv}}$, Item 1 of Theorem 1.6 is a special case of Theorem 11.1. Theorem 11.1 suggests that the exponential worst-case hardness of any problem that reduces to $\mathsf{NP_{sv}}$ via a randomized reduction implies the average-case hardness of NP. A "search version" of Theorem 11.1 can be proved in a similar way.

Reminder of Item 1 of Theorem 1.8. If there exists a $2^{n^{1-\delta}}$-time approximately size-verifiable function that cannot be inverted in time $2^{O(n/\log n)}$ in the worst case for some constant $\delta > 0$, then $\mathsf{coNP} \times \{\mathcal{U}, \mathcal{T}\} \not\subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ for every constant c.

Proof Sketch. We define $t(n) := 2^{n^{1-\delta}}$ and replace Corollary 8.12 with Theorem 8.9 and Proposition 8.15 in the proof of Theorem 11.1. □

Restatement of Item 2 of Theorem 1.8. For any constants $\delta > 0$, $k \in \mathbb{N}$, and c,

$$\mathsf{BPTIME}\left(2^{n^{1-\delta}}\right)^{\Sigma_k^{\mathrm{p}}} \not\subseteq \mathsf{DTIME}\left(2^{O(n/\log n)}\right) \quad \Longrightarrow \quad \Pi^{\mathrm{p}}_{k+1} \times \{\mathcal{U}, \mathcal{T}\} \not\subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}.$$

Proof. Assume that $\Pi^{\mathrm{p}}_{k+1} \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ for some constant c. By Theorem 7.1, every language in $\mathsf{BPP}^{\Sigma_k^{\mathrm{p}}}$ admits a universal heuristic scheme. By Corollary 6.5, we obtain

$$\mathsf{BPTIME}\left(2^{n^{1-\delta}}\right)^{\Sigma_k^{\mathrm{p}}} \subseteq \mathsf{DTIME}\left(2^{O(n/\log n)}\right).$$

□

Reminder of Item 3 of Theorem 1.8. For every constant $\delta > 0$,

$$\mathsf{AMTIME}\left(2^{n^{1-\delta}}\right) \not\subseteq \mathsf{DTIME}\left(2^{O(n/\log n)}\right) \implies \mathsf{NP} \times \{\mathcal{U}, \mathcal{T}\} \not\subseteq \mathsf{Avg}_\mathsf{P}\mathsf{P}.$$

Proof. Assume that $\mathsf{NP} \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{Avg}_\mathsf{P}\mathsf{P}$. By Theorem 10.1, every language $L \in \mathsf{NP}$ admits a universal heuristic scheme. By Lemma 7.10, every language in $\mathsf{BP} \cdot \mathsf{NP} = \mathsf{AM}$ admits a universal heuristic scheme. By Corollary 6.5, we obtain

$$\mathsf{AMTIME}\left(2^{n^{1-\delta}}\right) \subseteq \mathsf{DTIME}\left(2^{O(n/\log n)}\right)$$

$\square$

Reminder of Theorem 1.9. The following are equivalent:

1. $\mathsf{DistPH} \subseteq \mathsf{Avg}_\mathsf{P}\mathsf{P}$.
2. $\mathsf{DistPH} \subseteq \mathsf{AvgP}$.
3. $\mathsf{PH} \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ for some constant c.

Proof. It is evident that 1 $\implies$ 2 $\implies$ 3. To prove the converse, assume that $\mathsf{PH} \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ for some constant c. By Theorems 7.1 and 9.6, we obtain $\mathsf{Dist}\Sigma^\mathrm{p}_k \subseteq \mathsf{Dist}(\mathsf{BPP}^{\Sigma^\mathrm{p}_k}) \subseteq \mathsf{Avg}_\mathsf{P}\mathsf{P}$ for every constant $k \in \mathbb{N}$, from which it follows that $\mathsf{DistPH} = \bigcup_k \mathsf{Dist}\Sigma^\mathrm{p}_k \subseteq \mathsf{Avg}_\mathsf{P}\mathsf{P}$. $\square$

Restatement of Theorem 1.10. If $\mathsf{coNP} \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{Avg}^1_{1-n^{-c}}\mathsf{P}$ for some constant c, then $\mathsf{DistNP}_\mathsf{sv} \subseteq \mathsf{Avg}_\mathsf{P}\mathsf{P}$.

Proof. Under the assumption of the theorem, by Corollary 8.12, every language in NP_sv admits a universal heuristic scheme. By Theorem 9.6, we obtain $\mathsf{DistNP}_\mathsf{sv} = \mathsf{NP}_\mathsf{sv} \times \mathrm{PS}\textsc{amp} \subseteq \mathsf{Avg}_\mathsf{P}\mathsf{P}$. $\square$

We demonstrate that a sparse language yields better worst-case-to-average-case connections. As a concrete example, consider the parameterized version MCSP$[s]$ of MCSP for a parameter $s \colon \mathbb{N} \to \mathbb{N}$. MCSP$[s]$ is the language of the truth tables of $f \colon \{0,1\}^n \to \{0,1\}$ such that there is a circuit of size $s(n)$ that computes f. The following result strengthens[29] Corollary 1.7.

Theorem 11.2. *Let $s \colon \mathbb{N} \to \mathbb{N}$ be an efficiently computable function such that $n^{1.1} \leq s(n) \leq 2^n/n$. If* MCSP$[s] \notin \mathsf{DTIME}\left(2^{O(s(\log N))}\right)$, *then* $\mathsf{DistNP} \not\subseteq \mathsf{Avg}_\mathsf{P}\mathsf{P}$ *and* $\mathsf{Dist}\Sigma^\mathrm{p}_2 \not\subseteq \mathsf{AvgP}$.

A trivial brute-force algorithm for MCSP$[s]$ runs in time $s(\log N)^{O(s(\log N))}$ on inputs of length $N = 2^n$, and no better algorithm is known; thus, the hypothesis of Theorem 11.2 is plausible.

[29] Similar results hold for $\mathfrak{C}$-MCSP.

Proof of Theorem 11.2. Since circuits of size s can be encoded by $O(s \log s)$ bits, the number of Yes instances of length $N = 2^n$ in MCSP$[s]$ is bounded by $2^{O(s(n) \log s(n))} =: 2^{s'(N)}$. Applying either Corollary 9.10 or Theorem 7.1, we obtain a universal heuristic scheme for MCSP$[s] \in \mathsf{NP}$. It follows from Corollary 6.6 that

$$\mathrm{MCSP}[s] \in \mathsf{DTIME}\left(2^{O(s'(N)/\log(s'(N)/\log N))}\right) \subseteq \mathsf{DTIME}\left(2^{O(s(\log N))}\right).$$

$\square$

Finally, we mention that our results provide "NP-hardness" of GapMINKT$^{\mathsf{NP}}$.

Theorem 11.3. *For any constants $\delta > 0$,*

$$\mathsf{BPTIME}\left(2^{n^{1-\delta}}\right)^{\mathsf{NP}} \not\subseteq \mathsf{DTIME}\left(2^{O(n/\log n)}\right) \quad \Longrightarrow \quad \mathrm{GapMINKT}^{\mathsf{NP}} \notin \mathsf{P}$$

Proof. The proof of [Hir20a, Theorem III.7] shows that GapMINKT$^{\mathsf{NP}} \in \mathsf{P}$ if and only if Gap(K$^{\mathsf{NP}}$ vs K) $\in \mathsf{P}$. Combining this with Lemma 7.2, we obtain that GapMINKT$^{\mathsf{NP}} \in \mathsf{P}$ implies that every language in $\mathsf{BPP}^{\mathsf{NP}}$ admits a universal heuristic scheme, which implies

$$\mathsf{BPTIME}\left(2^{n^{1-\delta}}\right)^{\Sigma_k^{\mathrm{p}}} \subseteq \mathsf{DTIME}\left(2^{O(n/\log n)}\right)$$

by Corollary 6.5.

$\square$

12 Concluding Remarks and Open Questions

In this paper, we developed the theory of "meta-computational average-case complexity", which was initiated in [Hir20a], and demonstrated the power of the theory using several new concepts, such as P-computable average-case polynomial time and universal heuristic scheme. We believe that these concepts deserve further study. Several open questions are presented below.

The main open question is to prove that $\mathsf{NP} \not\subseteq \mathsf{DTIME}(2^{o(n)})$ implies $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$. Our results come close to resolving this question and suggest the following approaches:

1. Can we prove that NP is reducible to the task of inverting approximately size-verifiable one-way functions in the worst case? It is not difficult to show this under the assumption that $\mathsf{NP} \subseteq \mathsf{coAM}$ (see Appendix A); however, it remains an open problem to prove the NP-hardness of inverting approximately size-verifiable one-way function under a plausible assumption.

2. Does $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ imply $\mathsf{DistNP} \subseteq \mathsf{Avg_P P}$?

3. Does $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ imply $\mathsf{Dist}\Sigma_2^{\mathrm{p}} \subseteq \mathsf{AvgP}$?

A positive answer to either one of these questions can be combined with our results to establish the implication from $\mathsf{NP} \not\subseteq \mathsf{DTIME}(2^{o(n)})$ to $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$.

Another important open question is to prove an "infinitely often" version of our results. For example, can we prove that $\mathsf{PH} \not\subseteq \mathsf{i.o.DTIME}(2^{o(n)})$ implies $\mathsf{DistPH} \not\subseteq \mathsf{i.o.AvgP}$? In this setting, Viola's barrier comes into play. Recall that we overcome the barrier of Theorem 1.3 by exploiting the fact that a hardness amplification procedure is defined on all input lengths; in other words, it is crucial in our proofs that a hypothetical algorithm solves DistPH *almost everywhere*. Extending our results to the infinitely-often setting would require significantly new ideas to overcome Viola's barrier in a different way.

Another question is whether the exponential-time hypothesis (ETH; [IPZ01]) implies $\mathsf{DistPH} \not\subseteq \mathsf{AvgP}$. ETH implies that $\mathsf{NP} \not\subseteq \mathsf{DTIME}(2^{o(n/\log n)})$,[30] but falls slightly short of the time bound $2^{O(n/\log n)}$ of Theorem 1.6.

Can we extend our one-sided-error hardness results to two-sided-error versions? Can we base the existence of (average-case) one-way functions on the worst-case hardness of UP? Recent results of Liu and Pass [LP20] characterized the two-sided-error hardness of MINKT by the existence of one-way functions, which may be useful for investigating these questions.

Finally, the last question is whether we can apply meta-computational proof techniques to small complexity classes, such as $\mathsf{DistP} \not\subseteq \mathsf{AvgL}$. The question of whether DistP is easy on average for log-space algorithms was investigated in [BCGL92]; however, no non-trivial relationship between $\mathsf{DistP} \not\subseteq \mathsf{AvgL}$ and the worst-case hardness of P is known. A new insight is required to apply our proof techniques to DistP, as we exploit the power of DistNP or DistPH in an essential manner.

Acknowledgement

We thank Rahul Santhanam and Hanlin Ren for helpful discussion and thank anonymous reviewers for helpful comments. This work was supported by JST, ACT-I and PRESTO Grant Number JPMJPR2024, Japan.

References

[ABK+06] Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from Random Strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006.

[AD17] Eric Allender and Bireswar Das. Zero knowledge and circuit minimization. *Inf. Comput.*, 256:2–8, 2017.

[AF09] Luis Filipe Coelho Antunes and Lance Fortnow. Worst-Case Running Times for Average-Case Algorithms. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 298–303, 2009.

[AFvMV06] Luis Antunes, Lance Fortnow, Dieter van Melkebeek, and N. V. Vinodchandran. Computational depth: Concept and applications. *Theor. Comput. Sci.*, 354(3):391–404, 2006.

[AGGM06] Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. On basing one-way functions on NP-hardness. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 701–710, 2006.

[30] Note that an m-variable $O(m)$-clause $O(1)$-width CNF formula can be encoded as a binary string of length $O(m \log m)$.

[AGvM$^+$18] Eric Allender, Joshua A. Grochow, Dieter van Melkebeek, Cristopher Moore, and Andrew Morgan. Minimum Circuit Size, Graph Isomorphism, and Related Problems. *SIAM J. Comput.*, 47(4):1339–1372, 2018.

[AH19] Eric Allender and Shuichi Hirahara. New Insights on the (Non-)Hardness of Circuit Minimization and Related Problems. *TOCT*, 11(4):27:1–27:27, 2019.

[AHM$^+$08] Eric Allender, Lisa Hellerstein, Paul McCabe, Toniann Pitassi, and Michael E. Saks. Minimizing Disjunctive Normal Form Formulas and AC^0 Circuits Given a Truth Table. *SIAM J. Comput.*, 38(1):63–84, 2008.

[AK20] Yahav Alon and Michael Krivelevich. Finding a Hamilton cycle fast on average using rotations and extensions. *Random Struct. Algorithms*, 57(1):32–46, 2020.

[AR88] Eric Allender and Roy S. Rubinstein. P-Printable Sets. *SIAM J. Comput.*, 17(6):1193–1202, 1988.

[BB15] Andrej Bogdanov and Christina Brzuska. On Basing Size-Verifiable One-Way Functions on NP-Hardness. In *Proceedings of the Theory of Cryptography Conference (TCC)*, pages 1–6, 2015.

[BCGL92] Shai Ben-David, Benny Chor, Oded Goldreich, and Michael Luby. On the Theory of Average Case Complexity. *J. Comput. Syst. Sci.*, 44(2):193–219, 1992.

[BFL01] Harry Buhrman, Lance Fortnow, and Sophie Laplante. Resource-Bounded Kolmogorov Complexity Revisited. *SIAM J. Comput.*, 31(3):887–905, 2001.

[BFP05] Harry Buhrman, Lance Fortnow, and Aduri Pavan. Some Results on Derandomization. *Theory Comput. Syst.*, 38(2):211–227, 2005.

[BFS09] Harry Buhrman, Lance Fortnow, and Rahul Santhanam. Unconditional Lower Bounds against Advice. In *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, pages 195–209, 2009.

[BLvM05] Harry Buhrman, Troy Lee, and Dieter van Melkebeek. Language compression and pseudorandom generators. *Computational Complexity*, 14(3):228–255, 2005.

[BS07] Andrej Bogdanov and Muli Safra. Hardness Amplification for Errorless Heuristics. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 418–426, 2007.

[BT06a] Andrej Bogdanov and Luca Trevisan. Average-Case Complexity. *Foundations and Trends in Theoretical Computer Science*, 2(1), 2006.

[BT06b] Andrej Bogdanov and Luca Trevisan. On Worst-Case to Average-Case Reductions for NP Problems. *SIAM J. Comput.*, 36(4):1119–1159, 2006.

[Coo71] Stephen A. Cook. The Complexity of Theorem-Proving Procedures. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 151–158, 1971.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, 22(6):644–654, 1976.

[FF93] Joan Feigenbaum and Lance Fortnow. Random-Self-Reducibility of Complete Sets. *SIAM J. Comput.*, 22(5):994–1005, 1993.

[GL89] Oded Goldreich and Leonid A. Levin. A Hard-Core Predicate for all One-Way Functions. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 25–32, 1989.

[GLN11] Oded Goldreich, Leonid A. Levin, and Noam Nisan. On Constructing 1-1 One-Way Functions. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 13–25. 2011.

[GRS00] Oded Goldreich, Ronitt Rubinfeld, and Madhu Sudan. Learning Polynomials with Queries: The Highly Noisy Case. *SIAM J. Discrete Math.*, 13(4):535–570, 2000.

[GS86] Shafi Goldwasser and Michael Sipser. Private Coins versus Public Coins in Interactive Proof Systems. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 59–68, 1986.

[GS87] Yuri Gurevich and Saharon Shelah. Expected Computation Time for Hamiltonian Path Problem. *SIAM J. Comput.*, 16(3):486–502, 1987.

[GS88] Joachim Grollmann and Alan L. Selman. Complexity Measures for Public-Key Cryptosystems. *SIAM J. Comput.*, 17(2):309–335, 1988.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A Pseudorandom Generator from any One-way Function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[Hir18] Shuichi Hirahara. Non-black-box Worst-case to Average-case Reductions within NP. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 247–258, 2018.

[Hir20a] Shuichi Hirahara. Characterizing Average-Case Complexity of PH by Worst-Case Meta-Complexity. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 50–60, 2020.

[Hir20b] Shuichi Hirahara. Non-Disjoint Promise Problems from Meta-Computational View of Pseudorandom Generator Constructions. In *Proceedings of the Computational Complexity Conference (CCC)*, pages 20:1–20:47, 2020.

[Hir20c] Shuichi Hirahara. Unexpected hardness results for Kolmogorov complexity under uniform reductions. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 1038–1051, 2020.

[Hir20d] Shuichi Hirahara. Unexpected Power of Random Strings. In *Proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*, pages 41:1–41:13, 2020.

[HMX10] Iftach Haitner, Mohammad Mahmoody, and David Xiao. A New Sampling Protocol and Applications to Basing Cryptographic Primitives on the Hardness of NP. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 76–87, 2010.

[HOS18] Shuichi Hirahara, Igor Carboni Oliveira, and Rahul Santhanam. NP-hardness of Minimum Circuit Size Problem for OR-AND-MOD Circuits. In *Proceedings of the Computational Complexity Conference (CCC)*, pages 5:1–5:31, 2018.

[HW20] Shuichi Hirahara and Osamu Watanabe. On Nonadaptive Security Reductions of Hitting Set Generators. In *Proceedings of the Approximation, Randomization, and Combinatorial Optimization (APPROX/RANDOM)*, pages 15:1–15:14, 2020.

[IL89] Russell Impagliazzo and Michael Luby. One-way Functions are Essential for Complexity Based Cryptography (Extended Abstract). In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 230–235, 1989.

[IL90] Russell Impagliazzo and Leonid A. Levin. No Better Ways to Generate Hard NP Instances than Picking Uniformly at Random. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 812–821, 1990.

[Ila20a] Rahul Ilango. Connecting Perebor Conjectures: Towards a Search to Decision Reduction for Minimizing Formulas. In *Proceedings of the Computational Complexity Conference (CCC)*, pages 31:1–31:35, 2020.

[Ila20b] Rahul Ilango. Constant Depth Formula and Partial Function Versions of MCSP are Hard. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 424–433, 2020.

[Imp95] Russell Impagliazzo. A Personal View of Average-Case Complexity. In *Proceedings of the Structure in Complexity Theory Conference*, pages 134–147, 1995.

[Imp11] Russell Impagliazzo. Relativized Separations of Worst-Case and Average-Case Complexities for NP. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 104–114, 2011.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

[IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E Requires Exponential Circuits: Derandomizing the XOR Lemma. In *Proceedings of the Symposium on the Theory of Computing (STOC)*, pages 220–229, 1997.

[Kar72] Richard M. Karp. Reducibility Among Combinatorial Problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, pages 85–103, 1972.

[KC00] Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 73–79, 2000.

[Ko85] Ker-I Ko. On Some Natural Complete Operators. *Theor. Comput. Sci.*, 37:1–30, 1985.

[Ko91] Ker-I Ko. On the Complexity of Learning Minimum Time-Bounded Turing Machines. *SIAM J. Comput.*, 20(5):962–986, 1991.

[Kre88] Mark W. Krentel. The Complexity of Optimization Problems. *J. Comput. Syst. Sci.*, 36(3):490–509, 1988.

[KS04] Johannes Köbler and Rainer Schuler. Average-case intractability vs. worst-case intractability. *Inf. Comput.*, 190(1):1–17, 2004.

[Lev73] Leonid Anatolevich Levin. Universal sequential search problems. *Problemy Peredachi Informatsii*, 9(3):115–116, 1973.

[Lev86] Leonid A. Levin. Average Case Complete Problems. *SIAM J. Comput.*, 15(1):285–286, 1986.

[LP20] Yanyi Liu and Rafael Pass. On One-way Functions and Kolmogorov Complexity. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 1243–1254, 2020.

[LW95] Luc Longpré and Osamu Watanabe. On Symmetry of Information and Polynomial Time Invertibility. *Inf. Comput.*, 121(1):14–22, 1995.

[Mas79] William J Masek. Some NP-complete set covering problems. *Unpublished manuscript*, 1979.

[MX10] Mohammad Mahmoody and David Xiao. On the Power of Randomized Reductions and the Checkability of SAT. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 64–75, 2010.

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

[Ost91] Rafail Ostrovsky. One-Way Functions, Hard on Average Problems, and Statistical Zero-Knowledge Proofs. In *Proceedings of the Structure in Complexity Theory Conference*, pages 133–138, 1991.

[RR97] Alexander A. Razborov and Steven Rudich. Natural Proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.

[Sip83] Michael Sipser. A Complexity Theoretic Approach to Randomness. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 330–335, 1983.

[SSS95] Jeanette P. Schmidt, Alan Siegel, and Aravind Srinivasan. Chernoff-Hoeffding Bounds for Applications with Limited Independence. *SIAM J. Discrete Math.*, 8(2):223–250, 1995.

[STV01] Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom Generators without the XOR Lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001.

[Tho89] Andrew Thomason. A simple linear expected time algorithm for finding a hamilton path. *Discret. Math.*, 75(1-3):373–379, 1989.

[TUZ07] Amnon Ta-Shma, Christopher Umans, and David Zuckerman. Lossless Condensers, Unbalanced Expanders, And Extractors. *Combinatorica*, 27(2):213–240, 2007.

[TV07] Luca Trevisan and Salil P. Vadhan. Pseudorandomness and Average-Case Complexity Via Uniform Reductions. *Computational Complexity*, 16(4):331–364, 2007.

[Vad12] Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012.

[Vio05a] Emanuele Viola. On Constructing Parallel Pseudorandom Generators from One-Way Functions. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 183–197, 2005.

[Vio05b] Emanuele Viola. The complexity of constructing pseudorandom generators from hard functions. *Computational Complexity*, 13(3-4):147–188, 2005.

[VV86] Leslie G. Valiant and Vijay V. Vazirani. NP is as Easy as Detecting Unique Solutions. *Theor. Comput. Sci.*, 47(3):85–93, 1986.

[Wat12] Thomas Watson. Relativized Worlds without Worst-Case to Average-Case Reductions for NP. *TOCT*, 4(3):8:1–8:30, 2012.

[ZL70] AK Zvonkin and LA Levin. The complexity of finite objects and the algorithmic concepts of randomness and information. *UMN (Russian Math. Surveys)*, 25(6):83–124, 1970.

A NP-Completeness of $\mathsf{NP_{sv}}$ Under $\mathsf{NP} \subseteq \mathsf{coAM}$

We observe that $\mathsf{NP_{sv}}$ is NP-complete under the assumption that NP is easy for coAM algorithms.

Proposition A.1. *If* $\mathsf{NP} \subseteq \mathsf{coAM}$*, then* $\mathsf{NP_{sv}} = \mathsf{NP}$.

Proof. We will prove in Lemma A.2 that the assumption of Proposition A.1 implies that pr-AM = pr-coAM. Fix any $L \in \mathsf{NP}$, and let V be an NP-type verifier for L. Let Π^V be the promise problem defined in Definition 8.2. Then, using the lower bound protocol (Lemma 4.6), we have $\Pi^V \in$ pr-coAM = pr-AM, which implies that $L \in \mathsf{NP_{sv}}$. ☐

Lemma A.2. *If* $\mathsf{NP} \subseteq \mathsf{coAM}$*, then* pr-$\mathsf{AM} \subseteq$ pr-coAM.

Proof Sketch. Consider any promise problem $\Pi = (\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}}) \in$ pr-AM, and let V be a polynomial-time verifier for the AM protocol. This means that

1. if $x \in \Pi_{\mathrm{YES}}$, then for every r, for some y, $V(x, y, r) = 1$, and
2. if $x \in \Pi_{\mathrm{NO}}$, then $\Pr_r[\exists y, V(x, y, r) = 1] \le \frac{1}{4}$.

Let $L := \{ (x, r) \mid \exists y, V(x, y, r) = 1 \} \in \mathsf{NP}$. By the assumption, $L \in \mathsf{coAM}$, and thus there exists a polynomial-time verifier that witnesses $\mathsf{co}L \in \mathsf{AM}$ such that

1. if $(x, r) \in L$, then $\Pr_{r'}[\exists y', W(x, r, y', r') = 0] \le \frac{1}{4}$, and

2. if $(x, r) \notin L$, then for every r', for some y', $W(x, r, y', r') = 0$.

Combining these two conditions, we obtain

1. if $x \in \Pi_{\mathrm{YES}}$, then $\Pr_{r,r'}[\exists y', W(x, r, y', r') = 0] \leq \frac{1}{4}$, and
2. if $x \in \Pi_{\mathrm{NO}}$, then $\Pr_{r,r'}[\exists y', W(x, r, y', r') = 0] \geq \frac{3}{4}$,

which implies that $\Pi \in$ pr-coAM. $\square$

B Hamiltonian Path Admits P-Computable Average-Case Polynomial-Time Algorithms

Let $\mathcal{G}(n, p)$ denote the distribution of the Erdős-Rényi random graph in which each edge is included with probability p. For every function $p \colon \mathbb{N} \to [0, 1]$, let $\mathcal{G}_p := \{\mathcal{G}(n, p(n))\}_{n \in \mathbb{N}}$ be a family of the Erdős-Rényi random graphs. Let HamiltonianPath be the problem of deciding if a given graph contains a Hamiltonian path from the first vertex to the second vertex. We observe that the running time of the heuristic algorithm of Thomason [Tho89] is P-computable average-case polynomial-time.

Proposition B.1. *For every efficiently computable function p such that $1/p(n) \leq O(\log n)$,*

$$(\textsc{HamiltonianPath}, \mathcal{G}_p) \in \mathsf{Avg}_{\mathsf{P}}\mathsf{P}.$$

Proof Sketch. The heuristic algorithm A of [Tho89] consists of three algorithms: A_1, A_2, and A_3. The algorithm A_3 is a dynamic programming algorithm that solves HamiltonianPath in the worst case in time $2^{O(n)}$. Let $t_3(n)$ denote an upper bound of the running time of A_3. The algorithms A_1 and A_2 are errorless heuristic algorithms that either find a Hamiltonian path or report that there is no Hamiltonian path or fail. The algorithm A_1 runs in time $n^{O(1)}$, and the algorithm A_2 runs in time $2^{O(1/p(n))} + n^{O(1)} = n^{O(1)}$. Let $t_{12}(n)$ denote a polynomial upper bound of the running time of A_1 and A_2. The heuristic algorithm A for HamiltonianPath is defined as follows: Apply A_1. If A_1 fails, apply A_2. If A_2 fails, apply A_3.

We define a time bound $t \colon \{0, 1\}^* \to \mathbb{N}$ that upper-bounds the running time of A as follows. For every input graph G of n vertices, if either A_1 or A_2 succeeds on input G, then $t(G)$ is defined as $t_{12}(n)$; otherwise, $t(G) := t_3(n)$. Since A_1 and A_2 run in polynomial time, the time bound t is computable in time $n^{O(1)}$. Moreover, [Tho89] showed that A_1 and A_2 fail with probability at most 2^{-cn} over a random graph $G \sim \mathcal{G}(n, p(n))$ for some constant $c > 0$. Thus, $\mathbb{E}_{G \sim \mathcal{G}(n,p(n))}[t(G)^{\epsilon}] \leq n^{O(1)}$ for some constant $\epsilon > 0$ (in fact, for $\epsilon := 1$). Therefore, A is a P-computable average-case polynomial-time algorithm for HamiltonianPath. $\square$

There are AvgP algorithms that solve HamiltonianPath for a wider range of p.

Theorem B.2 ([AK20])**.** *For every efficiently computable function p such that $1/p(n) \leq \sqrt{n}/70$,*

$$(\textsc{HamiltonianPath}, \mathcal{G}_p) \in \mathsf{AvgP}.$$

Closing the gap between Proposition B.1 and Theorem B.2 is an interesting research question.