The Computational Complexity Column

by

Michal Koucký

Computer Science Institute, Charles University
Malostranské nám. 25, 118 00 Praha 1, Czech Republic
koucky@iuuk.mff.cuni.cz
https://iuuk.mff.cuni.cz/~koucky/

Meta-Computational Average-Case Complexity: A New Paradigm Toward Excluding Heuristica

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

Abstract

One of the central open questions in computational complexity theory is to connect worst-case hardness of NP to average-case hardness of NP. This question is well known as whether Heuristica is excluded from Impagliazzo's five worlds. It is known that standard proof techniques that relate worst-case and average-case complexities are incapable of excluding Heuristica; thus, in order to make progress, we need to develop new proof techniques.

Recently, new worst-case to average-case connections that cannot be proved by previous proof techniques are established based on *meta-complexity*. Meta-complexity refers to the computational complexity of problems that themselves ask complexity. In this article, we present the emerging paradigm of "meta-computational average-case complexity," i.e., a new approach of analyzing average-case complexity via meta-complexity of time-bounded Kolmogorov complexity.

1 Introduction

Traditionally, the complexity of a computational problem is measured in terms of worst-case complexity; that is, the performance of an algorithm is measured with respect to the worst input. However, it is often criticized that the worst-case complexity measure is too pessimistic: the worst input might not be encountered in practice; thus, worst-case complexity may not be relevant to "real-life" complexity. Moreover, some NP-complete problems, such as the Hamiltonian path problem, can be solved efficiently with high probability when the input is chosen randomly from natural distributions [GS87].

A better approach to measuring "real-life" complexity would be to use *average-case complexity*. It is natural to assume that a real-life instance is generated by some efficiently computable procedure. This motivates us to study the complexity of DistNP, i.e., the average complexity of NP over instances generated by some randomized polynomial-time algorithms.

The theory of average-case complexity has two primary motivations. As mentioned above, one is to understand the practical performance of algorithms. The other is that average-case hardness of NP is the foundation for the security of cryptosystems: the existence of an average-case hard problem in NP is necessary for most cryptographic primitives to be secure [IL89]. The relationship among $\mathsf{P} \neq \mathsf{NP}$, average-case hardness of NP, and cryptography was clearly presented in the influential work of Impagliazzo [Imp95]. We review Impagliazzo's five worlds in Section 2 and explain the importance of studying average-case complexity of NP.

Arguably, one of the most important open questions in the theory of average-case complexity is to exclude Heuristica from Impagliazzo's five worlds, i.e., to prove average-case hardness of NP from worst-case hardness of NP. This question can be, for example, formally stated as follows.

Open Question 1. Does $\mathsf{P} \neq \mathsf{NP}$ imply $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$?

Here, the statement $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$ means that there exists a pair of a problem L and a polynomial-time samplable distribution $\mathcal{D}$ with respect to which there is no average-polynomial-time algorithm that solves L. We review the basics of average-case complexity in Section 3, following the excellent survey of Bogdanov and Trevisan [BT06a].

There are at least three reasons why it is difficult to resolve Open Question 1. Standard proof techniques that connect worst-case complexity to average-case complexity, such as black-box reductions [FF93; BT06b], hardness amplification procedures [Vio05b; Vio05a], and relativizing proof techniques [Imp11; HN21], are shown to be incapable of resolving Open Question 1. (We will explain the details of the limits of black-box reductions in Section 5.) To make progress on the central problem, we need to develop new types of proof techniques that are not subject to any of these technical barriers.

Recently, a new type of proof techniques that are not subject to some of the barriers are developed. [Hir18] presented the first worst-case-to-average-case connection that goes beyond the limits of black-box reductions. Building on this result, [Hir21a] showed the following worst-case-to-average-case connections.

Theorem 2 ([Hir21a])**.**

1. *If* $\mathsf{UP} \not\subseteq \mathsf{DTIME}(2^{O(n/\log n)})$*, then* $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$.
2. *If* $\mathsf{PH} \not\subseteq \mathsf{DTIME}(2^{O(n/\log n)})$*, then* $\mathsf{DistPH} \not\subseteq \mathsf{AvgP}$.
3. *If* $\mathsf{NP} \not\subseteq \mathsf{DTIME}\left(2^{O(n/\log n)}\right)$*, then* $\mathsf{DistNP} \not\subseteq \mathsf{Avg_PP}$. Here, $\mathsf{Avg_PP}$ stands for P*-computable average-polynomial-time*, which interpolates P and AvgP; see Section 3 for a definition.

The first item of the worst-case-to-average-case connections of Theorem 2 cannot be proved by neither black-box reductions [FF93; BT06b] nor hardness amplification procedures [Vio05a; Vio05b]; any proof of Theorem 2 must simultaneously overcome the two barriers, i.e., the limits of black-box reductions and the "impossibility" of hardness amplification procedures. This is the reason why it took 35 years to find a proof of Theorem 2 since Levin [Lev86] laid the foundation of average-case complexity.

The main goal of this article is to present a proof of the first and second items of Theorem 2 as well as technical tools developed in the proof. Our presentation follows recent alternative proofs of [Hir21a], which are independently obtained by [Hir21b; GK22]. Along the way, we put together several results that are scattered in several papers.

The reader may wonder where the time complexity $2^{O(n/\log n)}$ comes from. It comes from the fact that a polynomial p is a "$(1/\epsilon \log n)$-exponential function"[1] for a small constant $\epsilon > 0$ in the sense that the $\epsilon \log n$-iterated composition $p^{(\epsilon \log n)}(n)$ of p is at most $2^{n/\log n}$. How this property is used in the proof will be explained in Section 8. In fact, the time complexity is shown to be tight for relativizing proof techniques: Building on the work of Impagliazzo [Imp11],

[1]The name is an analogue of a half-exponential function, which is a function f such that $f(f(n)) \leq 2^n$.

Hirahara and Nanashima [HN21] showed that there is an oracle A such that $\mathsf{DistPH}^A \subseteq \mathsf{AvgP}^A$ and $\mathsf{UP}^A \cap \mathsf{coUP}^A \not\subseteq \mathsf{DTIME}(2^{o(n/\log n)})^A$, which indicates that the time complexity $2^{O(n/\log n)}$ achieved in Theorem 2 cannot be improved to $2^{o(n/\log n)}$ using relativizing proof techniques. We note, however, that the first and third items of Theorem 2 is not necessarily relativizing; whether they can be relativized or not remains open,[2] whereas the second item of Theorem 2 does relativize.

Given the quantitatively tight barrier, it is natural to wonder if one can achieve the time complexity $2^{o(n/\log n)}$. Chen, Hirahara, and Vafa [CHV22] achieved this in fine-grained complexity settings. For example, they showed that nondeterministic linear time $\mathsf{NTIME}(n)$ cannot be solved in quasi-linear time on average if $\mathsf{UP} \not\subseteq \mathsf{DTIME}\left(2^{O(\sqrt{n\log n})}\right)$. The conclusion is weaker than Theorem 2, but the worst-case hardness assumption is also significantly weakened. A high-level idea of [CHV22] is that a quasi-linear function $p(n) = \widetilde{O}(n)$ is a "$\sqrt{\log n/n}$-exponential function" in the sense that $p^{(\sqrt{n/\log n})}(n) \leq 2^{O(\sqrt{n\log n})}$.

The proof is based on meta-complexity of time-bounded Kolmogorov complexity. *Meta-complexity* refers to the computational complexity of problems that themselves ask complexity. One representative example of meta-computational problems is MINKT [Ko91], which is the problem of computing the t-time-bounded Kolmogorov complexity $\mathrm{K}^t(x)$ of x, given $(x, 1^t)$ as input. Throughout this article, time-bounded Kolmogorov complexity and the (meta-)complexity of MINKT play a central role. We review these notions in Section 4. At a very high level, the reason why meta-complexity is useful is that worst-case meta-complexity exactly characterizes the average-case complexity of the polynomial hierarchy PH: [Hir20a] showed that $\mathsf{DistPH} \subseteq \mathsf{AvgP}$ if and only if $\mathrm{GapMINKT}^{\mathsf{PH}} \in \mathsf{P}$, where $\mathrm{GapMINKT}^{\mathsf{PH}}$ is the problem of approximating PH-oracle time-bounded Kolmogorov complexity in the worst case. Meta-complexity enables us to analyze average-case complexity from a view point of worst-case complexity. Analyzing worst-case complexity is often easier than analyzing average-case complexity. Indeed, by going through the statements on worst-case meta-complexity (see the right half of Fig. 1), new statements on average-case complexity of PH are proved in [Hir20a; Hir21a]: One-sided-error heuristics that succeed with small probability, errorless heuristics that succeed with high probability, average-polynomial-time algorithms (AvgP), and P-computable average-polynomial-time algorithms ($\mathsf{Avg}_{\mathsf{P}}\mathsf{P}$) are all equivalent for DistPH.

The aforementioned work [Hir18] showed the equivalence between the average-case complexity of MINKT and the worst-case complexity of an approximate version of MINKT. We present this result in Section 5 and explain why meta-complexity enables overcoming the limits of black-box reductions. Along the way, we introduce the notion of *k-wise direct product generator* [Hir20c], which is the main technical tool extensively used throughout this article.

For those who are familiar with randomness extractor, it may be useful to contrast the recent development of meta-complexity with the development of the theory of randomness extractor. The influential work of Trevisan [Tre01] presented a generic construction of an extractor from a pseudorandom generator construction. The k-wise direct product generator is also a (very simple) pseudorandom generator construction. Trevisan [Tre01] exploited the property of a pseudorandom generator construction *information-theoretically* to construct an extractor, which is an information-theoretic object. In contrast, the recent development of meta-complexity is given by exploiting the property of a pseudorandom generator construction *computationally*. In a bit more detail, we apply the reconstruction property of a pseudorandom generator construction

[2]The only non-relativizing part is Theorem 3, which relies on the PCP theorem.

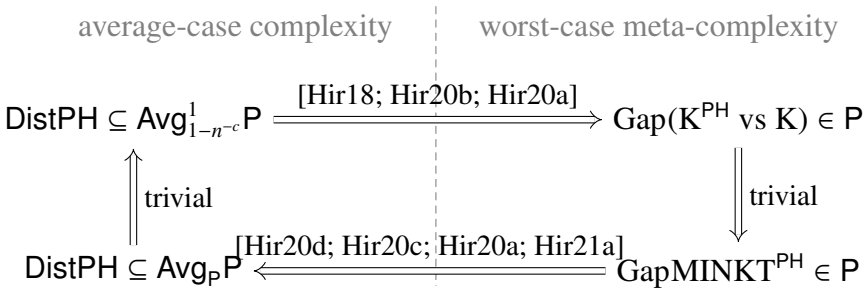

Figure 1: The paradigm of analyzing average-case complexity by worst-case meta-complexity. The figure shows that the following are equivalent for DistPH: Avg$^1_{1-n^{-c}}$P, i.e., one-sided-error heuristics that succeed with probability n^{-c}; AvgP, i.e., errorless heuristic schemes, or equivalently, average-polynomial-time algorithms; Avg$_P$P, i.e., P-computable average-polynomial-time algorithms. This equivalence is given as a corollary of the characterization of the average-case complexities of PH by worst-case meta-complexity of GapMINKT^{PH}.

(Lemma 12) to *efficient* algorithms, whereas Trevisan [Tre01] applied it to *inefficient* algorithms.

Two fundamental theorems of Kolmogorov complexity play a key role in the proof of Theorem 2: Language compression and symmetry of information. These theorems are known to be true unconditionally for *resource-unbounded* Kolmogorov complexity [ZL70]. We present *time-bounded* analogues of language compression and symmetry of information in Sections 6 and 7, respectively, under the assumption that DistNP ⊆ AvgP. The proofs of these results are "meta-computational" in the sense that we use an efficient hypothetical algorithm that computes time-bounded Kolmogorov complexity; this means that these proofs must be significantly different from the resource-unbounded cases, as resource-unbounded Kolmogorov complexity cannot be computed in finite steps.

The outline of the proof of Theorem 2 is depicted in Fig. 2. In Section 8, we introduce the notion of universal heuristic scheme. We construct universal heuristic schemes for PH in Section 9. As alluded in the figure, the final result does not refer to meta-complexity at all, whereas the proof goes through statements on worst-case meta-complexity. Indeed, the only known proofs are via meta-complexity!

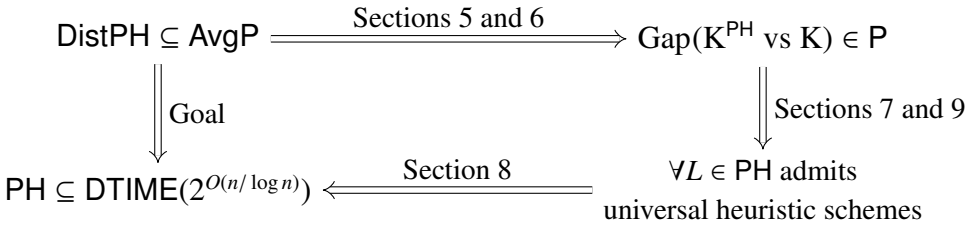

Figure 2: A proof strategy for the second item of Theorem 2.

Notation $[n]$ denotes $\{1, \ldots, n\}$. $|x|$ denotes the length of a string $x \in \{0,1\}^*$. For a function $p\colon \mathbb{N} \to \mathbb{N}$ and $i \in \mathbb{N}$, the function $p^{(i)}\colon \mathbb{N} \to \mathbb{N}$ is recursively defined as follows: $p^{(0)}(n) := n$ and $p^{(i+1)}(n) := p^{(i)}(p(n))$ for every $n \in \mathbb{N}$.

2 Impagliazzo's Five Worlds

What is the importance of the P versus NP question? One cannot underestimate significant impacts that a proof of P = NP would have. An efficient algorithm for NP would have great impacts on mathematics in general. If P = NP, for any mathematical theorem φ, one can find a proof π for φ in time $\mathsf{poly}(|\pi|, |\varphi|)$. This means that mathematicians do not need to work hard to find proofs. In a world where P = NP, the only task of mathematicians is to come up with the statements of interesting theorems; then, they can use the efficient automated theorem proving algorithm to find proofs of the theorems. Another consequence of P = NP is that the security of any public-key cryptosystem can be broken. This does not just mean that the privacy of communications is compromised. All the wealth invested to Bitcoin can be stolen by the person who finds a proof of P = NP, as Bitcoin relies on the security of a public-key cryptosystem.[3]

However, most researchers conjecture that P ≠ NP, so it is likely that the answer to the P versus NP question is negative. Given that most researchers believe P ≠ NP, it is more intriguing to investigate what follows from P ≠ NP. In principle, we can expect that a secure public-key cryptosystem can be constructed; however, currently, there is a large gap between P ≠ NP and the possibility of cryptography, which was clearly explained by Impagliazzo [Imp95].

Impagliazzo [Imp95] proposed five possible worlds which are consistent with the current knowledge of complexity theory and named them as follows: Algorithmica, Heuristica, Pessiland, Minicrypt, and Cryptomania. Algorithmica is a world in which NP is easy in the worst case, e.g., P = NP. Heuristica is a world in which there exists an efficient heuristic for NP, i.e., NP is easy on average, but NP is hard in the worst case. For example, one canonical definition of Heuristica is a world in which P ≠ NP and DistNP ⊆ AvgP.[4] Pessiland is a world in which NP is hard on average and there is no one-way function; this is the worst of all the possible worlds. The existence of one-way functions is often considered as a minimal assumption under which complexity-theory-based cryptography is possible [IL89]. In Pessiland, cryptography is not possible and NP cannot be solved efficiently on average. Minicrypt is a world in which one-way functions exist but a public-key cryptosystem does not exist. Finally, Cryptomania is a world in which a public-key cryptosystem exists. A popular conjecture is that our world is Cryptomania. The security of the RSA cryptosystem, which is widely used in practice, is not yet broken despite significant efforts by many researchers; thus, it is reasonable to conjecture that the RSA cryptosystem is secure (at least against classical computers [Sho99]) and, in particular, our world is Cryptomania.

What is known about Impagliazzo's five worlds? It is easy to see the following four implications: ∃ a public-key cryptosystem $\Longrightarrow$ ∃ a one-way function $\Longrightarrow$ DistNP $\not\subseteq$ AvgP $\Longrightarrow$ NP ≠ P $\Longrightarrow$ True. The converses of these implications correspond to the open problems of excluding Minicrypt, Pessiland, Heuristica, and Algorithmica, respectively. Excluding these worlds is equivalent to showing that our world is Cryptomania, i.e., a secure public-key cryptosystem exists; excluding any one of them is one of the most important open questions in complexity theory and cryptography. Currently, no world is excluded from Impagliazzo's five worlds. The difficulty is that there are many technical barriers suggesting that standard proof techniques would not work. For example, to prove P ≠ NP (i.e., to exclude Algorithmica from

[3] It is worth mentioning that Bitcoin also relies on a cryptographic primitive called *Proof of Work* [DN92], which is based on average-case hardness of NP; see, e.g., [BRSV17; BRSV18; HS21].

[4] There are many variants of Heuristica, depending on what types of algorithms we consider. For example, a world in which NP $\not\subseteq$ BPP and DistNP ⊆ AvgBPP can be considered as a variant of Heuristica.

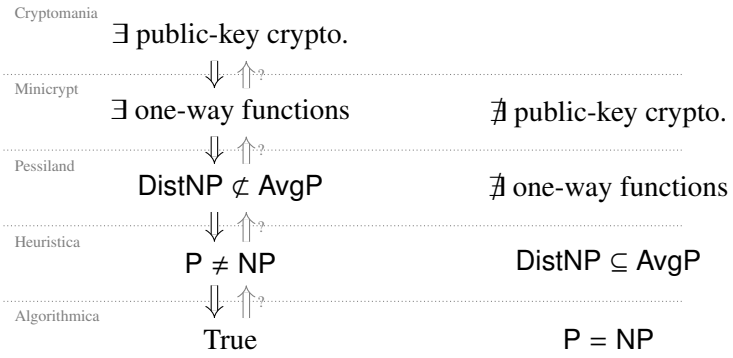

Figure 3: Impagliazzo's five worlds. "⇒" indicates known implications. "$\stackrel{?}{\Rightarrow}$" indicates open questions, each of which corresponds to excluding Algorithmica, Heuristica, Pessiland, and Minicrypt.

the possible worlds), one needs to develop a proof technique that simultaneously overcomes the relativization barrier [BGS75], the algebrization barrier [AW09], the natural proof barrier [RR97], and the locality barrier [CHOPRS20]. Similarly, to exclude Heuristica, one needs to develop a proof technique that simultaneously overcomes the limits of black-box reductions [FF93; BT06b; AGGM06; BB15], the "impossibility" of hardness amplification procedures [Vio05b; Vio05a], and the relativization barrier [Wat12; Imp11; HN21]. In this article, we present a proof technique that is not subject to the first two barriers but is still subject to the relativization barrier; overcoming the relativization barrier is left open.

3 Average-Case Complexity

The theory of average-case complexity studies a *distributional problem* $(L, \mathcal{D})$, which is a pair of a language $L \subseteq \{0,1\}^*$ and a family $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ of distributions over instances of "size" n. For example, we define the uniform distribution $\mathcal{U}$ to be the family $\{\mathcal{U}_n\}_{n\in\mathbb{N}}$ such that $\mathcal{U}_n$ is the uniform distribution over $\{0,1\}^n$. In this specific example, all the strings in the support of $\mathcal{U}_n$ are binary strings of length n; we call such a family of distributions *length-preserving*; however, we emphasize that there is no restriction on $\mathcal{D}_n$ in general, except that $\mathcal{D}_n$ is a distribution over $\{0,1\}^*$.[5]

The seminal work of Levin [Lev86] introduced a robust notion of average-case easiness. An algorithm A is said to be an *average-polynomial-time algorithm for* $(L, \mathcal{D})$ if $A(x, 1^n) = L(x)$ for every $x \in \{0,1\}^*$ and there exists some constant $\epsilon > 0$ such that for all large $n \in \mathbb{N}$, it holds that $\mathbb{E}_{x\sim\mathcal{D}_n}[t_A(x,1^n)^\epsilon] \le n^{O(1)}$, where $t_A(x,1^n)$ is an upper bound on the running time of an algorithm A on input $(x,1^n)$.[6] The class of distributional problems for which there exist average-polynomial-time algorithms is denoted by AvgP. When there exists an average-case-polynomial upper bound $t_A\colon \{0,1\}^* \to \mathbb{N}$ on the running time of A such that t_A is computable in polynomial time (in the worst case), we say that A is P-*computable average-polynomial-time*; the class of distributional problems solvable by P-computable average-polynomial-time algorithms is

[5] One restriction that automatically follows from $\mathcal{D} \in \mathrm{PSAMP}$ is that the length of any string in the support of $\mathcal{D}_n$ is at most $n^{O(1)}$.

[6] We identify a language $L \subseteq \{0,1\}^*$ with its characteristic function $L\colon \{0,1\}^* \to \{0,1\}$.

denoted by $\mathsf{Avg_P P}$ [Hir21a].

One may wonder why the constant ϵ in the definition of average-polynomial-time is not fixed to 1. If $\epsilon = 1$, the definition just says that the expected running time $\mathbb{E}_{x\sim\mathcal{D}_n}[t_A(x, 1^n)]$ of an algorithm is bounded by a polynomial in the size parameter n, which appears to be a natural definition at first glance. A short answer to this question is that having a constant exponent $\epsilon > 0$ in the definition makes average-case hardness results stronger: $(L, \mathcal{D}) \notin \mathsf{AvgP}$ implies that $(L, \mathcal{D})$ cannot be solved by average-polynomial-time algorithms in the naïve definition.

Another reason is that AvgP is equivalent to another (perhaps more) intuitive notion: *errorless heuristic scheme* [Imp95; BT06a]. A polynomial-time algorithm A is said to be an *errorless heuristic scheme* if for every $(n, \delta^{-1}) \in \mathbb{N}$,

$$\Pr_{x\sim\mathcal{D}_n}\left[A(x, 1^n, 1^{\delta^{-1}}) \neq L(x)\right] \leq \delta$$

and $A(x, 1^n, 1^{\delta^{-1}}) \in \{L(x), \perp\}$ for every $x \in \mathrm{supp}(\mathcal{D}_n)$, where $\perp$ indicates a failure of A, i..e., the algorithm does not know the answer; that is, A is allowed to *fail* (i.e., output $\perp$) but is not allowed to *err* (i.e., output the wrong answer $1 - L(x)$). For a function $\delta\colon \mathbb{N} \to [0, 1]$, the class of distributional problems that can be solved by errorless heuristics with failure probability $\delta(n)$ on instances of size n is denoted by $\mathsf{Avg}_\delta\mathsf{P}$.

What distributions should we consider? A family of distributions $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ is said to be *polynomial-time samplable* if there exists a randomized polynomial-time algorithm M such that the distribution of $M(1^n)$ is identical to $\mathcal{D}_n$ for every $n \in \mathbb{N}$. The class of polynomial-time samplable distributions is denoted by PSamp. We define DistNP to be the class of distributional problems $(L, \mathcal{D})$ such that $L \in \mathsf{NP}$ and $\mathcal{D} \in$ PSamp.

It will be useful to consider a family $\mathcal{D} = \{\mathcal{D}_{a,b}\}_{a,b\in\mathbb{N}}$ of distributions indexed by a pair $(a, b) \in \mathbb{N}^2$ of integers. Such a family can be converted into a family $\mathcal{D}' = \{\mathcal{D}'_n\}_{n\in\mathbb{N}}$ indexed by a single integer n by defining $\mathcal{D}'_{\langle a,b\rangle} := \mathcal{D}_{a,b}$ for every $(a, b) \in \mathbb{N}^2$, where $\langle\text{-},\text{-}\rangle\colon \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ is a bijection defined as, for example, $\langle a, b\rangle := \sum_{i=0}^{a+b} i + a$.

Buhrman, Fortnow, and Pavan [BFP05] showed an important theorem that will be useful throughout this article. They showed pr-BPP = pr-P in Heuristica, i.e., randomized polynomial-time algorithms can be derandomized in polynomial time. The underlying tool of derandomization is the notion of *pseudorandom generator*. A function $G = \left\{G_n : \{0, 1\}^{s(n)} \to \{0, 1\}^n\right\}_{n\in\mathbb{N}}$ is said to be a *pseudorandom generator* secure against linear-sized circuits if for all large $n \in \mathbb{N}$, for every circuit D of size n,

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

Note that the existence of a pseudorandom generator with seed length $s(n)$ enables derandomizing randomized polynomial-time algorithms in time $2^{O(s(n)+\log n)}$; more background on derandomization can be found in [Vad12].

Theorem 3 (Buhrman, Fortnow, and Pavan [BFP05])**.** *If* $\mathsf{DistNP} \subseteq \mathsf{AvgP}$*, 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. Theorem 3 is based on the following four results.

1. $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ implies $\mathsf{NE} = \mathsf{E}$ [BCGL92].
2. $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ implies pr-MA = pr-NP [KS04].
3. If $\mathsf{NE} = \mathsf{E}$ and pr-MA = pr-NP, then $\mathsf{E} \not\subset \mathsf{i.o.SIZE}(2^{\epsilon n})$ for some constant $\epsilon > 0$ [BFP05]. (Proof Sketch: If not, we obtain $\mathsf{E} \subseteq \mathsf{i.o.MATIME}(2^{o(n)}) = \mathsf{i.o.NTIME}(2^{o(n)})$ using a PCP theorem. However, this contradicts the time hierarchy and $\mathsf{NE} = \mathsf{E}$; see [Hir20a] for the details.)
4. If $\mathsf{E} \not\subset \mathsf{i.o.SIZE}(2^{\epsilon n})$ for some constant $\epsilon > 0$, then there exists a polynomial-time-computable pseudorandom generator secure against linear-sized circuits [IW97].

□

4 Meta-Complexity of Time-Bounded Kolmogorov Complexity

4.1 Kolmogorov Complexity

The Kolmogorov complexity $\mathrm{K}(x)$ of a string $x \in \{0,1\}^*$ is defined to be the minimum length of a program that prints x. For example, $\mathrm{K}(1^n) = \log n + O(1)$ because the string 1^n can be described by a program "`print '1' n times`", whose description length is $\log n + O(1)$. Note that an integer $n \in \mathbb{N}$ can be represented as a binary string of length $\lceil \log n \rceil$ and the length of the other part of the program is constant.

The t-time-bounded Kolmogorov complexity $\mathrm{K}^t(x)$ of x is defined to be the minimum length of a program that prints x in time t. We also consider the conditional Kolmogorov complexity $\mathrm{K}^t(x \mid y)$ of x given y, which is the minimum length of a program that prints x given y as input in time t. More generally, we consider oracle time-bounded Kolmogorov complexity:

Definition 4 (Time-bounded Kolmogorov complexity)**.** *For strings $x, y \in \{0,1\}^*$, a time bound $t \in \mathbb{N} \cup \{\infty\}$, and an oracle A, the A-*oracle *t-*time-bounded Kolmogorov complexity of x given y *is defined as*

$$\mathrm{K}^{t,A}(x \mid y) := \min\left\{|M| \;\middle|\; M^A \text{ outputs } x \text{ in time } t\right\}.$$

Here, $|M|$ denotes the length of a binary encoding of an oracle Turing machine M.[7] *We omit the superscript A if $A = \emptyset$, the superscript t if $t = \infty$, and "$\mid y$" if y is the empty string.*

The Kolmogorov complexity of an n-bit string $x \in \{0,1\}^n$ satisfies $0 \le \mathrm{K}(x) \le n + O(1)$ because any string x can be described by a program "`print` x". The upper bound is almost tight:

Fact 5. *For any integer $s \ge 1$, the number of strings $x \in \{0,1\}^*$ such that $\mathrm{K}(x) < s$ is less than 2^s. In particular, $\Pr_{x \sim \{0,1\}^n}[\mathrm{K}(x) \ge s] \ge 1 - 2^{s-n}$.*

[7] More formally, we fix an efficient universal Turing machine U such that for every oracle Turing machine M, there exists a string $d_M \in \{0,1\}^*$ such that $U^A(d_M, x) = M^A(x)$ for every input $x \in \{0,1\}^*$ and every oracle $A \subseteq \{0,1\}^*$. Then, we define $|M|$ to be the length $|d_M|$ of d_M.

Proof. For every string x such that $\mathrm{K}(x) < s$, there is a program M_x of length less than s that prints x. The map $x \mapsto M_x$ is injective. The number of programs of length less than s is at most $\sum_{i=0}^{s-1} 2^i < 2^s$, so is the number of strings x such that $\mathrm{K}(x) < s$. □

More background on Kolmogorov complexity can be found in the book of Li and Vitányi [LV19].

4.2 Meta-Complexity

Here, we review the notion of meta-complexity that is relevant to Theorem 2. For broader background on meta-complexity and its recent development, see the survey of Allender [All21].

We first introduce the representative meta-computational problem: MINKT [Ko91] is defined to be the language

$$\mathrm{MINKT} := \{(x, 1^t, 1^s) \mid \mathrm{K}^t(x) \leq s\}.$$

This is a problem in NP because given a program M as certificate one can check whether $|M| \leq s$ and M prints x in time t. Its NP-completeness is a long-standing open question [Ko91]. Note that MINKT is computationally equivalent to the problem of computing the t-time-bounded Kolmogorov complexity $\mathrm{K}^t(x)$ of x on input $(x, 1^t)$; in other words, MINKT asks to compute the time-bounded Kolmogorov complexity of x. The complexity of MINKT is referred to *meta-complexity*, as MINKT itself asks for the complexity of printing an input x.

Although it is unknown whether MINKT can be solved *exactly* in Heuristica, we show in the next section that an *approximate* version of MINKT, denoted by GapMINKT, can be solved in Heuristica. To introduce the problem formally, we recall the framework of promise problems [Gol06]. A *promise problem* Π is a pair $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ of languages. An algorithm A is said to *solve* a promise problem $\Pi = (\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ if A accepts every $x \in \Pi_{\mathrm{YES}}$ and rejects every $x \in \Pi_{\mathrm{NO}}$. Note that if there exists an algorithm that solves a promise problem $\Pi = (\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$, it must be disjoint, i.e., $\Pi_{\mathrm{YES}} \cap \Pi_{\mathrm{NO}} = \emptyset$. It is common to require a promise problem to be disjoint in the definition; however, we also consider "non-disjoint" promise problems [Hir20b], i.e., promise problems that are non-disjoint under plausible assumptions, which will be useful in Section 6.

Definition 6 ([Ko91; Hir20b])**.** *For a polynomial $\tau\colon \mathbb{N} \to \mathbb{N}$ and an oracle $A \subseteq \{0,1\}^*$, let*

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

Then, we define

- $\mathrm{Gap}_\tau\mathrm{MINKT}^A := (\Pi^A_{\mathrm{YES}}, \Pi^A_{\mathrm{NO}})$ *and*
- $\mathrm{Gap}_\tau(\mathrm{K}^A \text{ vs } \mathrm{K}) := (\Pi^A_{\mathrm{YES}}, \Pi^\emptyset_{\mathrm{NO}})$.

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 every $A \in \mathfrak{C}$. We omit the superscript A if $A = \emptyset$.*

For example, $\mathrm{GapMINKT}^{\mathrm{SAT}}$ is the problem of approximating SAT-oracle time-bounded Kolmogorov complexity. The problem $\mathrm{Gap}(\mathrm{K}^{\mathrm{SAT}} \text{ vs } \mathrm{K})$ is computationally harder than $\mathrm{GapMINKT}^{\mathrm{SAT}}$ and is an example of "non-disjoint" promise problems: It is non-disjoint if $\mathsf{E}^{\mathsf{NP}} \neq \mathsf{E}$ [Hir20b].

Nevertheless, we show in Section 6 that Gap($\mathrm{K}^{\mathrm{SAT}}$ vs K) can be solved in polynomial time if $\mathsf{DistPH} \subseteq \mathsf{AvgP}$.

One common misunderstanding about GapMINKT$^{\mathrm{SAT}}$ is that it should be NP-hard as SAT is NP-complete. In fact, proving NP-hardness of GapMINKT$^{\mathrm{SAT}}$ would significantly improve Theorem 2: $\mathsf{P} \neq \mathsf{NP}$ implies $\mathsf{DistPH} \not\subseteq \mathsf{AvgP}$ if GapMINKT$^{\mathrm{SAT}}$ is NP-hard. Thus, it is an important open problem to prove the NP-hardness of GapMINKT$^{\mathrm{SAT}}$, despite the intuition that approximating SAT-oracle time-bounded Kolmogorov complexity seems much harder than computing SAT.

The problem GapMINKT can be equivalently defined as the problem of computing an approximate value of $\mathrm{K}^t(x)$ up to some additive error.

Fact 7 ([Hir21a])**.** *The following are equivalent.*

1. GapMINKT $\in \mathsf{P}$.
2. *There exist a polynomial-time algorithm* $\widetilde{\mathrm{K}}$ *and a polynomial* p *such that*
$$\mathrm{K}^{p(t)}(x) - \log p(t) \leq \widetilde{\mathrm{K}}(x, 1^t) \leq \mathrm{K}^t(x)$$
for every string $x \in \{0,1\}^*$ *and every integer* $t \geq |x|$.

Proof Sketch. Given a polynomial-time algorithm M that solves GapMINKT, the algorithm $\widetilde{\mathrm{K}}$ can be defined as follows:
$$\widetilde{\mathrm{K}}(x, 1^t) := \min\{s \in \mathbb{N} \mid M(x, 1^t, 1^s) = 1\}.$$
□

Fact 7 shows that GapMINKT is equivalent to the problem of computing $\mathrm{K}^t(x)$ up to an additive error of
$$\mathrm{K}^t(x) - (\mathrm{K}^{p(t)}(x) - \log p(t)) = \mathrm{cd}^{t,p(t)}(x) + \log p(t),$$
where $\mathrm{cd}^{t,s}(x) := \mathrm{K}^t(x) - \mathrm{K}^s(x)$ is called (s,t)*-time-bounded computational depth*. This notion plays a key role in Section 8.

Definition 8 (Time-Bounded Computational Depth; [AFMV06; Hir21a])**.** *For time bounds* $s \in \mathbb{N}$ *and* $t \in \mathbb{N} \cup \{\infty\}$ *and a string* $x \in \{0,1\}^*$*, the* (s,t)-time-bounded computational depth *of* x *is defined as*
$$\mathrm{cd}^{s,t}(x) := \mathrm{K}^s(x) - \mathrm{K}^t(x).$$
We omit the superscript t *if* $t = \infty$.

The notion of (time-unbounded) computational depth $\mathrm{cd}^t(\text{-})$ was introduced by Antunes, Fortnow, Melkebeek, and Vinodchandran [AFMV06]. One fundamental property of computational depth is that it is small for most strings. For example, [AFMV06] showed that $\mathrm{cd}^{\mathsf{poly}(n)}(x) \leq k$ holds with probability at least $1 - 2^{-k+O(\log n)}$ over a random input x drawn from a *polynomial-time-computable distribution* $\mathcal{D}$. Here, a polynomial-time-computable distribution is a distribution whose cumulative function can be computed in polynomial time. Under a plausible assumption, Antunes and Fortnow [AF09] generalized it to polynomial-time-*samplable* distributions; [Hir21a] proved the same result in Heuristica.

Theorem 9 ([AF09; Hir21a])**.** *Assume either* $\mathsf{E} \not\subseteq \mathsf{i.o.DSPACE}(2^{\epsilon n})$ *for some constant* $\epsilon > 0$ *or* $\mathsf{DistNP} \subseteq \mathsf{AvgP}$*. Then, for every* $\mathcal{D} \in \mathrm{PSAMP}$*, there exists a polynomial* t *such that for every* $n \in \mathbb{N}$*,*
$$\Pr_{x \sim \mathcal{D}_n}\left[\mathrm{cd}^{t(n)}(x) > k\right] \leq 2^{-k+\log t(n)}.$$

5 Non-Black-Box Worst-Case-to-Average-Case Reduction

Feigenbaum and Fortnow [FF93] and Bogdanov and Trevisan [BT06b] presented fundamental limits of proof techniques called *(black-box) reductions*. A natural approach to showing a worst-case-to-average-case connection would be to construct a reduction. In order to show the implication from the worst-case hardness of a language L to the average-case hardness of a distributional problem $(L', \mathcal{D})$, it suffices to design a reduction R that takes an arbitrary oracle O that solves $(L', \mathcal{D})$ on average and decides L in the worst case. Typically, the correctness of a reduction can be shown regardless of the complexity of an oracle O; i.e., R^O decides L for *every* oracle $O \subseteq \{0,1\}^*$. Such a reduction R is called *black-box*. Bogdanov and Trevisan [BT06b] showed the following limits of black-box reductions.

Theorem 10 ([BT06b])**.** *Let L be any language outside* $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$. *Then, there is no randomized polynomial-time nonadaptive reduction from L to* DistNP.

To exclude Heuristica, it suffices to show that some NP-complete problem is reducible to DistNP. NP-complete problems are believed to be outside $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$, as otherwise PH collapses [Yap83]. Theorem 10 indicates that, unless PH collapses, NP-complete problems cannot be reducible to DistNP via a randomized polynomial-time nonadaptive reduction.

How can we overcome this limits? One approach is to consider *adaptive* reductions. There are adaptive reductions in the literature (e.g., [HILL99; MR07]) and there exists an artificial language that admits an adaptive random-self-reduction but not nonadaptive one [FFLS94]. However, it is unknown if there exists an adaptive reduction for a natural language that goes beyond the limits $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$ of black-box reductions. The other approach is to use *non-black-box* reductions. Here, we say that a reduction R is non-black-box if the reduction exploits an efficiency of an oracle, i.e., R^O may fail to decide L correctly if O cannot be decided in polynomial time.[8]

We now demonstrate that GapMINKT admits a worst-case-to-average-case connection. The connection is proved by a non-black-box reduction that exploits the efficiency of an oracle, i.e., an efficient hypothetical algorithm that solves DistNP.

Theorem 11 ([Hir18; Hir20b])**.** *If* $\mathsf{DistNP} \subseteq \mathsf{AvgP}$, *then* $\mathrm{GapMINKT} \in \mathsf{P}$.

To prove this theorem, we introduce a *k-wise direct product generator* [Hir20c], which turned out to be a fundamental tool for analyzing Kolmogorov complexity. A k-wise direct product generator $\mathrm{DP}_k \colon \{0,1\}^n \times \{0,1\}^{nk} \to \{0,1\}^{nk+k}$ is defined as follows:

$$\mathrm{DP}_k(x; z) := (z^1, \ldots, z^k, \langle z^1, x\rangle, \ldots, \langle z^k, x\rangle)$$

for every $x \in \{0,1\}^n$ and every $z = (z^1, \ldots, z^k) \in (\{0,1\}^n)^k$, where $\langle x, y\rangle$ denotes the inner product of x and $y \in \{0,1\}^n$ over GF(2), i.e., $\langle x, y\rangle := (\sum_{i=1}^n x_i y_i) \mod 2$. The k-wise direct product generator $\mathrm{DP}_k(x; \text{-})$ is a pseudorandom generator secure against an algorithm D if $\mathrm{K}(x \mid D) > k + O(\log |x|)$. More formally, we have the following property:

Lemma 12 (Deterministic Reconstruction for DP_k; see [Hir21a])**.** *Assume that* $\mathsf{DistNP} \subseteq \mathsf{AvgP}$. *Then, there exists 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*

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

[8] In fact, more importantly, O should not depend on the input x, which is used in the proof of Theorem 11.

where $z \sim \{0,1\}^{nk}$, $w \sim \{0,1\}^{nk+k}$, *and* $r \sim \{0,1\}^{s}$, *it holds that*

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

This lemma can be proved by standard proof techniques of pseudorandomness [Vad12, Chapter 7] together with Theorem 3. We briefly present a proof sketch.

Proof Sketch. Using Yao's next-bit predictor, given some prefix of $\mathrm{DP}_k(x; z)$, the next bit can be predicted. Note that the first nk bits of $\mathrm{DP}_k(x; z)$ are completely uniform and cannot be predicted. Thus, there exists some index $i \in [k]$ such that $\langle z^i, x\rangle$ can be predicted with probability at least $\frac{1}{2} + \frac{\epsilon}{k}$, given $z^1, \ldots, z^k, \langle z^1, x\rangle, \ldots, \langle z^{i-1}, x\rangle$ as input. Note that $\langle z^i, x\rangle$ is the z^i-th bit of the Hadamard code of x. Using the list-decoding algorithm of Goldreich and Levin [GL89], one can decode x from the next-bit predictor. We have described a randomized algorithm M that prints x given $z^1, \ldots, z^k, \langle z^1, x\rangle, \ldots, \langle z^{i-1}, x\rangle$ as input. However, the random bits $z^1, \ldots, z^k$ as well as the internal randomness of M can be generated from the pseudorandom generator G of Theorem 3; i.e., there exists a short seed $\sigma \in \{0,1\}^{O(\log n)}$ from which $z^1, \ldots, z^k$ can be generated. Now we are ready to describe a program M' that prints x: Given a seed σ and an "advice string" $\alpha := (\langle z^1, x\rangle, \ldots, \langle z^{i-1}, x\rangle) \in \{0,1\}^{i-1}$, the program M' computes $(z^1, \ldots, z^k) := G(\sigma)$ and outputs the output of M on input $z^1, \ldots, z^k, \langle z^1, x\rangle, \ldots, \langle z^{i-1}, x\rangle$. The length of the program M' is approximately at most $|\sigma| + |\alpha| \leq k + O(\log n)$. □

In the literature of pseudorandom generator constructions and randomness extractors [TV07; TUZ07; Uma09], the length of the advice string α is referred to as the *advice complexity* of the pseudorandom generator construction $\mathrm{DP}_k(x; \text{-})$. Trevisan and Vadhan [TV07] showed that the advice complexity of any pseudorandom generator construction that extends its seed by k bits must be at least $k - 3$; thus, the simple pseudorandom generator construction $\mathrm{DP}_k(x; \text{-})$ achieves the optimal advice complexity up to an additive logarithmic term.[9]

Given Lemma 12, the proof of Theorem 11 is fairly simple.

Proof of Theorem 11. Consider a family $\mathcal{D} := \{\mathcal{D}_{n,t}\}_{n,t\in\mathbb{N}}$ of distributions over instances of MINKT such that $\mathcal{D}_{n,t}$ is the distribution defined by the following sampling procedure: Pick $x \sim \{0,1\}^n$ uniformly at random and output $(x, 1^t, 1^{n-2})$.

Since $(\mathrm{MINKT}, \mathcal{D}) \in \mathsf{DistNP} \subseteq \mathsf{AvgP}$, there exists an errorless polynomial-time heuristic M that solves $(\mathrm{MINKT}, \mathcal{D})$ with failure probability at most $\frac{1}{4}$. We now present a randomized algorithm R that solves GapMINKT: The algorithm R takes $(x, 1^t, 1^s)$ as input and accepts if and only if $M(\mathrm{DP}_k(x; z), 1^{t'}) \in \{1, \perp\}$ for a random $z \sim \{0,1\}^{nk}$, where $n := |x|$, k and $t' = \mathsf{poly}(t, n)$ are parameters chosen later.

To see the correctness of R, consider any YES instance $(x, 1^t, 1^s)$ of GapMINKT; i.e., $\mathrm{K}^t(x) \leq s$. Since the string $\mathrm{DP}_k(x; z)$ can be described from the seed z and a program that prints x in time t, we have

$$\mathrm{K}^{t'}(\mathrm{DP}_k(x; z)) \leq |z| + \mathrm{K}^t(x) + O(\log n) \leq nk + s + O(\log n) \leq nk + k - 2$$

[9] In the original paper [Hir18], the pseudorandom generator of Theorem 3 is not used; in this case, to obtain a small approximation error in Theorem 11, one needs to use a pseudorandom generator construction that has small *randomness complexity* in addition to small advice complexity. Whether the approximation error of [Hir18] can be improved or not under the assumption that MINKT is easy on average (under which the existence of a pseudorandom generator is unknown) remains open.

for a sufficiently large t', where we choose a large $k = s + O(\log n)$ so that the last inequality holds. This ensures that $(\mathrm{DP}_k(x; z), 1^{t'}, 1^{nk+k-2})$ is a YES instance of MINKT for every z. By the property of an errorless heuristic scheme, M outputs either 1 or $\perp$ on input $(\mathrm{DP}_k(x; z), 1^{t'}, 1^{s'})$; hence, R accepts with probability 1 over a choice of $z \sim \{0,1\}^{nk}$.

Conversely, we claim that R rejects any No instance $(x, 1^t, 1^s)$ of $\mathrm{Gap}_\tau\mathrm{MINKT}$ with high probability over the internal randomness of R (i.e., the random choice $z \sim \{0,1\}^{nk}$), where τ is a polynomial chosen later. Intuitively, this is because an algorithm M cannot distinguish the output distribution $\mathrm{DP}_k(x; \text{-})$ of the k-wise direct product generator from the uniform distribution if $\mathrm{K}^{\mathsf{poly}(t)}(x \mid M) > k + O(\log n) = s + O(\log n)$. More formally, we prove the contrapositive: Assume that R accepts $(x, 1^t, 1^s)$ with probability at least $\frac{7}{8}$ over a choice of z; that is,

$$\Pr_z\left[M(\mathrm{DP}_k(x; z), 1^{t'}) \in \{1, \perp\}\right] \geq \frac{7}{8}.$$

We claim that $(x, 1^t, 1^s)$ is *not* a No instance of $\mathrm{Gap}_\tau\mathrm{MINKT}$ for some polynomial τ. If we pick $w \sim \{0,1\}^{nk+k}$, then by Fact 5, with probability at least $\frac{1}{2}$, it holds that $\mathrm{K}(w) > |w| - 2$, in which case $(w, 1^{t'}, 1^{|w|-2})$ is a No instance of MINKT; hence,

$$\Pr_w\left[M(w, 1^{t'}) \in \{1, \perp\}\right] \leq \Pr_w\left[M(w, 1^{t'}) = \perp\right] + \Pr_w\left[(w, 1^{t'}, 1^{|w|-2}) \in \mathrm{MINKT}\right] \leq \frac{1}{4} + \frac{1}{2} = \frac{3}{4}.$$

Let D be a circuit such that $D(w) := 1$ iff $M(w, 1^{t'}) \in \{1, \perp\}$. Then, it follows from the two inequalities above that

$$\Pr_z\left[D(\mathrm{DP}_k(x; z)) = 1\right] - \Pr_w\left[D(w) = 1\right] \geq \frac{7}{8} - \frac{3}{4} = \frac{1}{8}.$$

By Lemma 12, we obtain

$$\mathrm{K}^{p(t,n)}(x \mid D) \leq k + \log p(t, n)$$

for some polynomial p. Since the circuit D can be described using $O(\log t')$ bits, we conclude that

$$\mathrm{K}^{\tau(t,n)}(x) \leq s + \log \tau(t, n) \tag{1}$$

for some polynomial τ, which means that $(x, 1^t, 1^s)$ is not a No instance of $\mathrm{Gap}_\tau\mathrm{MINKT}$.

We conclude that R is a one-sided-error randomized polynomial-time algorithm for $\mathrm{Gap}_\tau\mathrm{MINKT}$; that is, $\mathrm{Gap}_\tau\mathrm{MINKT} \in \text{pr-}\mathsf{coRP} \subseteq \text{pr-}\mathsf{BPP} = \text{pr-}\mathsf{P}$, where the last equality follows from Theorem 3. □

Why is the reduction R constructed in the proof of Theorem 11 non-black-box? The key point is that Eq. (1) may not hold when there is no polynomial-time algorithm M that decides MINKT. Specifically, the circuit D comes from the hypothetical efficient algorithm M that solves MINKT. It would be possible to show that $\mathrm{K}^{\tau(t,n),\mathrm{MINKT}}(x) \leq s + \log \tau(t, n)$ unconditionally; however, removing the MINKT oracle seems to be impossible in general. In fact, under plausible conjectures, the reduction R must be non-black-box. For example, the conjecture by Rudich [Rud97] implies that GapMINKT $\notin \mathsf{coNP/poly}$; thus, if R were non-black-box, then we would get a contradiction to the limits of black-box reductions (Theorem 10). Other evidences that Theorem 11 is inherently non-black-box can be found in [Hir18; HW20].

It is instructive to emphasize that meta-complexity plays a key role in constructing the non-black-box reduction. The problem GapMINKT is a meta-computational problem that asks

the complexity of generating a given string x in a given time limit t. It is this meta-computational property of GapMINKT that enables us to exploit the efficiency of a hypothetical algorithm M in the proof of Theorem 11.

We mention that the proof techniques of Theorem 11 actually show the equivalence between the worst-case hardness of GapMINKT and the existence of a polynomial-time-computable hitting set generator. A family $H = \left\{H_n : \{0,1\}^{s(n)} \to \{0,1\}^n\right\}_{n\in\mathbb{N}}$ is said to be a *hitting set generator* secure against $\mathsf{i.o.P}$ if for every polynomial-time algorithm M, there exist infinitely many integers $n \in \mathbb{N}$ such that $\Pr_{x\sim\{0,1\}^n}[M(x) = 1] \geq \frac{1}{2}$ implies $M(H_n(z)) = 1$ for some seed $z \in \{0,1\}^{s(n)}$.

Proposition 13 ([Hir20a; Hir21b])**.** *Under the assumption that* $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}(2^{\epsilon n})$ *for some constant* $\epsilon > 0$*, the following are equivalent.*

1. $\mathrm{GapMINKT} \in \mathsf{P}$.
2. *There exists a constant c such that no polynomial-time-computable hitting set generator*

$$H = \left\{H_n : \{0,1\}^{n-c\log n} \to \{0,1\}^n\right\}_{n\in\mathbb{N}}$$

   is secure against $\mathsf{i.o.P}$.
3. *There exists a constant c such that* $(\mathrm{MINKT}, \mathcal{U}^*) \in \mathsf{Avg}_{\frac{1}{4}}\mathsf{P}$*, where* $\mathcal{U}^* = \left\{\mathcal{U}^*_{n,t}\right\}_{n,t\in\mathbb{N}}$ *is the family of distributions* $\mathcal{U}^*_{n,t}$ *that sample* $(x, 1^t, 1^{n-c\log n})$ *for* $x \sim \{0,1\}^n$.
4. *For every length-preserving distribution* $\mathcal{D} \in \mathrm{PSAMP}$*, there exists a polynomial* t_0 *such that for every* $t \geq t_0$*, there exists an errorless heuristic scheme that computes* $\mathrm{K}^{t(|x|)}(x)$ *over a random instance x chosen from* $\mathcal{D}$.

Although we include a proof sketch for completeness, the reader may skip the proof.

Proof Sketch. Item 2 $\Rightarrow$ 1: The idea is to use a "universal" hitting set generator $H = \{H_n\}$. Let H_n be the function that takes as input a program M of length less than $n - c\log n$ (which can be encoded as a binary string of length $n - c\log n$) and outputs the output of M if M halts in time n^2. Clearly, H_n is computable in $\mathsf{poly}(n)$. The image of H_n contains the set $X_n := \left\{x \in \{0,1\}^n \;\middle|\; \mathrm{K}^{n^2}(x) < n - c\log n\right\}$. Since H is not secure against $\mathsf{i.o.P}$, there exists a polynomial-time algorithm M that rejects every $x \in X_n$ and accepts at least a half of all the n-bit strings. Then, GapMINKT can be solved by the following algorithm R: Given $(x, 1^t, 1^s)$ as input, R picks $z \sim \{0,1\}^{nk}$ and $w \sim \{0,1\}^t$ randomly and outputs $1 - M(\mathrm{DP}_k(x; z) \cdot w)$, where $n := |x|$ and $k := s + O(\log nt)$. Here, w is used for padding the input of M so that the time bound $|\mathrm{DP}_k(x; z) \cdot w|^2$ is sufficiently larger than t. The correctness can be proved in a way similar to Theorem 11. See [Hir20a, Theorem 8.7] for the details.

Item 1 $\Rightarrow$ 3: Let M be a polynomial-time algorithm that solves $\mathrm{Gap}_\tau\mathrm{MINKT}$ for some polynomial τ. Then, an errorless heuristic M' for $(\mathrm{MINKT}, \mathcal{U}^*)$ can be defined as follows: $M'(x, 1^t, 1^s) := \perp$ if $M(x, 1^t, 1^s) = 1$; otherwise, $M'(x, 1^t, 1^s) := 0$, where s is fixed to $n - c\log n$. To see the correctness of M', observe that M' can err only on YES instances of MINKT If $(x, 1^t, 1^s) \in \mathrm{MINKT}$, then we have $M(x, 1^t, 1^s) = 1$, which implies that M' outputs $\perp$; thus, M' does not err. The probability that M' outputs $\perp$ is bounded by the probability that $M(x, 1^t, 1^s) = 1$, which happens only if $(x, 1^t, 1^s)$ is not a No instance of $\mathrm{Gap}_\tau\mathrm{MINKT}$. It follows from Fact 5

that $\mathrm{K}(x) \geq n - 2 > s + \log \tau(n,t) = n - c \log n + \log \tau(n,t)$ with probability at least $\frac{3}{4}$ for a large constant c; hence, the probability that M' fails is at most $\frac{1}{4}$.

Item 3 $\Rightarrow$ 2: Let c be a constant chosen later. Let H be an arbitrary family

$$H = \left\{ H_n : \{0,1\}^{n - c \log n} \to \{0,1\}^n \right\}_{n \in \mathbb{N}},$$

which is computable in time $\ll t(n)$. Given an errorless heuristic M for $(\mathrm{MINKT}, \mathcal{U}^*)$, we define M' so that $M'(x) := 0$ if $M(x, 1^{t(n)}, 1^{n - c' \log n}) \in \{1, \perp\}$ and $M'(x) := 1$ otherwise, for every $x \in \{0,1\}^n$. We show that M' "avoids" H:

1. The algorithm M' accepts at least a half of $\{0,1\}^n$ because at least a $(1 - o(1))$-fraction of $\{0,1\}^n$ is a No instance of MINKT and the failure probability of M is at most $\frac{1}{4}$.

2. We claim that M' rejects every string x in the image of H_n: Note that $x \in \{0,1\}^n$ can be described by an integer $n \in \mathbb{N}$ and a seed $z \in \{0,1\}^{n - c \log n}$ such that $H_n(z) = x$, which implies that $\mathrm{K}^{t(n)}(x) \leq n - c \log n + O(\log n) \leq n - c' \log n$ for a sufficiently large c; this means that $(x, 1^{t(n)}, 1^{n - c' \log n}) \in \mathrm{MINKT}$, which is rejected by M'.

Item 1 $\Leftrightarrow$ 4: This is proved in [Hir21b] using SoI of Section 7. We omit the proof. □

We conclude this section by describing applications to derandomization. Given that a hitting set generator is used to derandomize randomized algorithms [GVW11], it is natural to wonder if one can characterize *complexity-theoretic* hitting set generators using the proof techniques of Proposition 13. Here, a hitting set generator or a pseudorandom generator $G = \left\{ G : \{0,1\}^{s(n)} \to \{0,1\}^n \right\}_{n \in \mathbb{N}}$ with seed length $s(n)$ is said to be *complexity-theoretic* if it can be computed in time $2^{O(s(n) + \log n)}$. This notion is more appropriate for derandomization [NW94] and is weaker than *cryptographic* generators. For example, the pseudorandom generator of Theorem 3 is complexity-theoretic. In [Hir20b], the existence of complexity-theoretic hitting set generators is characterized by the worst-case hardness of approximating Levin's Kt complexity [Lev84], which is an exponential-time variant of time-bounded Kolmogorov complexity. We refer the reader to [Hir20b] for details.

6 Algorithmic Language Compression

The language compression theorem [Sip83; BFL01; BLM05] for resource-unbounded Kolmogorov complexity refers to the following simple and fundamental fact:

Fact 14 (Language Compression)**.** *Let L be a decidable language. Then, for every $n \in \mathbb{N}$ and every $x \in \{0,1\}^n \cap L$,*

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

Proof Sketch. Any string $x \in L \cap \{0,1\}^n$ can be described by the index of x in the enumeration of $L \cap \{0,1\}^n$ and an integer $n \in \mathbb{N}$. □

Sipser [Sip83] considered a time-bounded analogue of the language compression theorem and showed that $\mathrm{K}^{\mathsf{poly}(n),\mathrm{SAT}}(x \mid r) \leq |L \cap \{0,1\}^n| + O(\log n)$ for $x \in L \in \mathsf{P}$, where r is a random string. Here, we show that the SAT oracle and the random string r can be removed if $\mathsf{DistNP} \subseteq \mathsf{AvgP}$. At a very high level, the idea is that the SAT oracle can be replaced with

an errorless heuristic for DistNP. We also remove the random string using Theorem 3. It will be useful to state the language compression theorem for a family of languages indexed by an integer t, which we call an *ensemble of languages*:

Definition 15 (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 $|x| \leq p_L(t)$ 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 language compression theorem can be stated as follows.

Theorem 16 (Time-Bounded Language Compression [Hir21a])**.** *Let A be an oracle and $L = \{L_t\}_{t\in\mathbb{N}} \in \mathsf{NP}^A$ be an ensemble of languages. If $\mathsf{DistNP}^A \subseteq \mathsf{AvgP}$, then there exists a polynomial p such that*

$$\mathrm{K}^{p(t)}(x) \leq \log|L_t| + \log p(t)$$

for every $(x, 1^t) \in L$.

We present an "algorithmic" proof of the language compression theorem, which generalizes the non-black-box reduction of Theorem 11. Instead of just showing Theorem 16, we construct a polynomial-time algorithm that accepts $x \in L_t$ and rejects any string x such that $\mathrm{K}^{\mathsf{poly}(t)}(x) > \log|L_t| + O(\log t)$. Formally:

Theorem 17 (Algorithmic Language Compression [Hir21a])**.** *Let A be an oracle and $L = \{L_t\}_{t\in\mathbb{N}} \in \mathsf{NP}^A$ be an ensemble of languages. If $\mathsf{DistNP}^A \subseteq \mathsf{AvgP}$, then there exists a polynomial p such that a promise problem $\Pi = (\Pi_{\mathrm{Yes}}, \Pi_{\mathrm{No}})$ defined as*

$$\Pi_{\mathrm{Yes}} := \left\{(x, 1^t, 1^k) \;\middle|\; x \in L_t\right\},$$
$$\Pi_{\mathrm{No}} := \left\{(x, 1^t, 1^k) \;\middle|\; \mathrm{K}^{p(t)}(x) > k + \log p(t), k \geq \log|L_t| + 1\right\}$$

is in pr-P.

The language compression theorem follows from the disjointness of Π_{Yes} and Π_{No}.

Proof of Theorem 16 from Theorem 17. The existence of an algorithm that separates Π_{Yes} from Π_{No} implies that $\Pi_{\mathrm{Yes}} \cap \Pi_{\mathrm{No}} = \emptyset$. Consider any string $x \in L_t$. Let $k := \log|L_t| + 1$. Since $(x, 1^t, 1^k) \in \Pi_{\mathrm{Yes}}$, we obtain $(x, 1^t, 1^k) \notin \Pi_{\mathrm{No}}$, which implies that $\mathrm{K}^{p(t)}(x) \leq k + \log p(t) = \log|L_t| + 1 + \log p(t)$. □

Now we present the algorithmic proof of the language compression theorem. The proof is quite similar to Theorem 11: For an errorless heuristic B for some distributional problem in DistNP^A, we consider a randomized non-black-box reduction B' that outputs 1 on input x iff $B(\mathrm{DP}_k(x; z)) \in \{1, \perp\}$ for a random choice of z.

Proof of Theorem 17. Let $L' = \left\{(\mathrm{DP}_k(x; z), 1^n, 1^t, 1^k) \;\middle|\; x \in L_t \cap \{0,1\}^n\right\}$. We claim that $L' \in \mathsf{NP}^A$. Note that $(w, 1^n, 1^t, 1^k) \in L'$ if and only if there exist $x \in \{0,1\}^n$, a certificate y for $x \in L_t$, and $z \in \{0,1\}^{n\cdot k}$ such that $w = \mathrm{DP}_k(x; z)$, which can be verified in polynomial time; thus, $L' \in \mathsf{NP}^A$. Consider a distribution $\mathcal{D} = \{\mathcal{D}_{n,t,k}\}_{n,t,k\in\mathbb{N}}$ such that $\mathcal{D}_{n,t,k}$ picks $w \sim \{0,1\}^{nk+k}$ and outputs $(w, 1^n, 1^t, 1^k)$. Since $(L', \mathcal{D}) \in \mathsf{DistNP}^A \subseteq \mathsf{AvgP}$, there exists an errorless heuristic B that solves L' with failure probability at most $\frac{1}{4}$.

We first claim that the number of YES instances in L' is relatively small. For each n, t, and $k \in \mathbb{N}$, we have

$$\begin{aligned}
&\Pr_{w\sim\{0,1\}^{nk+k}}\left[(w, 1^n, 1^t, 1^k) \in L'\right] \\
&= \Pr_{w\sim\{0,1\}^{nk+k}}\left[\exists x \in L_t \cap \{0,1\}^n, \exists z \in \{0,1\}^{nk}, w = \mathrm{DP}_k(x; z)\right] \\
&\leq |L_t \cap \{0,1\}^n| \cdot 2^{nk} \cdot 2^{-nk-k} \leq |L_t| \cdot 2^{-k}. \qquad (2)
\end{aligned}$$

We present a randomized algorithm B' that solves the promise problem Π defined in Theorem 17. On input $(x, 1^t, 1^k)$, the algorithm B' lets $n := |x|$ and picks $z \sim \{0,1\}^{nk}$ randomly, and accepts if and only if $B(\mathrm{DP}_k(x; z), 1^n, 1^t, 1^k) \in \{1, \perp\}$. We claim the correctness of B' below. Let p_L be the polynomial of Definition 15.

Claim 18. *For all large $t \in \mathbb{N}$ and every $n \leq p_L(t)$, the following hold for every $x \in \{0,1\}^n$.*

1. *If $(x, 1^t, 1^k) \in \Pi_{\mathrm{YES}}$, then $\Pr_z[B'(x, 1^t, 1^k; z) = 1] = 1$.*
2. *If $(x, 1^t, 1^k) \in \Pi_{\mathrm{NO}}$, then $\Pr_z[B'(x, 1^t, 1^k; z) = 1] < \frac{7}{8}$.*

Proof. Assume that $x \in L_t$. By the definition of L', we have $(\mathrm{DP}_k(x; z), 1^n, 1^t, 1^k) \in L'$ for every z. Since B is an errorless heuristic, we obtain $B(\mathrm{DP}_k(x; z), 1^n, 1^t, 1^k) \in \{1, \perp\}$, which implies that $B'(x, 1^t, 1^k; z) = 1$. This completes the proof of Item 1.

To prove Item 2 by way of contradiction, we assume that $\Pr_z[B'(x, 1^t, 1^k; z) = 1] \geq \frac{7}{8}$; that is,

$$\Pr_z\left[B(\mathrm{DP}_k(x; z), 1^n, 1^t, 1^k) \in \{1, \perp\}\right] \geq \frac{7}{8}. \qquad (3)$$

On the other hand,

$$\begin{aligned}
\Pr_w\left[B(w, 1^n, 1^t, 1^k) \in \{1, \perp\}\right] &\leq \Pr_w\left[L'(w, 1^n, 1^t, 1^k) = 1\right] + \Pr_w\left[B(w, 1^n, 1^t, 1^k) = \perp\right] \\
&\leq |L_t| \cdot 2^{-k} + \frac{1}{4}, \qquad (4)
\end{aligned}$$

where the last inequality uses Eq. (2). Since $(x, 1^t, 1^k) \in \Pi_{\mathrm{NO}}$, we have $|L_t| \cdot 2^{-k} \leq \frac{1}{2}$. By Eqs. (3) and (4), we obtain

$$\Pr_z\left[B(\mathrm{DP}_k(x; z), 1^n, 1^t, 1^k) \in \{1, \perp\}\right] - \Pr_w\left[B(w, 1^n, 1^t, 1^k) \in \{1, \perp\}\right] \geq \frac{7}{8} - \left(\frac{1}{2} + \frac{1}{4}\right) = \frac{1}{8}.$$

By Lemma 12, we obtain $\mathrm{K}^{p(t)}(x) \leq k + \log p(t)$ for some polynomial p. This is a contradiction to the assumption that $(x, 1^t, 1^k) \in \Pi_{\mathrm{NO}}$. ◇

It follows from Claim 18 that $\Pi \in$ pr-coRP. Using Theorem 3, we conclude that $\Pi \in$ pr-BPP = pr-P. □

The algorithmic language compression theorem generalizes the non-black-box reduction of Theorem 11. By algorithmically compressing an ensemble of languages $L_{t,s} := \{x \in \{0,1\}^* \mid \mathrm{K}^t(x) \leq s\}$, it can be shown that GapMINKT $\in$ P. More generally:

Corollary 19 ([Hir20b])**.** *Let A be an oracle. If* $\mathsf{DistNP}^A \subseteq \mathsf{AvgP}$*, then* $\mathrm{Gap}(\mathrm{K}^A \text{ vs } \mathrm{K}) \in \mathsf{P}$.

Proof. Consider an ensemble $L = \{L_{\langle n,t,s\rangle}\}_{n,t,s\in\mathbb{N}}$ of languages defined as[10]

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

Observe that $|L_{\langle n,t,s\rangle}| \le 2^{s+1}$ by Fact 5. It is easy to observe that $L \in \mathsf{NP}^A$. Applying Theorem 17 to L, we obtain a polynomial-time algorithm that solves the promise problem $(\Pi_{\text{Yes}}, \Pi_{\text{No}})$ such that

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

for some polynomial p. $\mathrm{Gap}_\tau(\mathrm{K}^A \text{ vs } \mathrm{K})$ is reducible to this promise problem via the reduction that maps $(x, 1^t, 1^s)$ to $(x, 1^{\langle |x|,t,s\rangle}, 1^{s+2})$ for some polynomial τ. □

The following lemma shows that any string that can be efficiently compressed with some PH oracle can also be compressed without the oracle if $\mathsf{DistPH} \subseteq \mathsf{AvgP}$.

Lemma 20 ([Hir20b])**.** *Let A be an oracle. If $\mathrm{Gap}(\mathrm{K}^A \text{ vs } \mathrm{K}) \in \mathsf{P}$, then there exists a polynomial p such that, for every $x \in \{0,1\}^*$ and every $t \ge |x|$,*

$$\mathrm{K}^{p(t)}(x) \le \mathrm{K}^{t,A}(x) + \log p(t).$$

Proof. Let $(\Pi_{\text{Yes}}, \Pi_{\text{No}}) := \mathrm{Gap}_\tau(\mathrm{K}^A \text{ vs } \mathrm{K})$. The hypothesis implies that $\Pi_{\text{Yes}} \cap \Pi_{\text{No}} = \emptyset$. Since $(x, 1^t, 1^s) \in \Pi_{\text{Yes}}$ for $s := \mathrm{K}^{t,A}(x)$, we obtain $(x, 1^t, 1^s) \notin \Pi_{\text{No}}$, which implies that $\mathrm{K}^{\tau(|x|,t)}(x) \le s + \log\tau(|x|,t) = \mathrm{K}^{t,A}(x) + \log\tau(|x|,t)$. □

7 Symmetry of Information

Yet another fundamental theorem of Kolmogorov complexity is *symmetry of information*, which was established by Kolmogorov and Levin [ZL70]. It states that for every $x \in \{0,1\}^*$ and $y \in \{0,1\}^*$,

$$\mathrm{K}(x \mid y) + \mathrm{K}(y) \lesssim \mathrm{K}(x,y) \lesssim \mathrm{K}(y \mid x) + \mathrm{K}(x),$$

where "$\lesssim$" indicates that the inequality holds up to an additive logarithmic term. Note that the second inequality is trivial because the pair of strings (x,y) can be computed by combining a program of size $\mathrm{K}(x)$ that prints x with a program of size $\mathrm{K}(y \mid x)$ that prints y given x as input. The highly non-trivial part of symmetry of information is the first inequality. Here, we consider a time-bounded analogue of symmetry of information.

Definition 21. Symmetry of information for time-bounded Kolmogorov complexity (SoI) *refers to the following hypothesis: There exists a polynomial p such that for any strings $x \in \{0,1\}^*$ and $y \in \{0,1\}^*$, for every $t \ge |x| + |y|$,*

$$\mathrm{K}^{p(t)}(x \mid y) + \mathrm{K}^{p(t)}(y) - \log p(t) \le \mathrm{K}^t(x,y). \tag{SoI}$$

[10] We include n in the parameter so that L is an ensemble of languages.

Longpré and Watanabe [LW95] showed that $\mathsf{P} = \mathsf{NP}$ implies SoI, and that SoI implies the non-existence of one-way functions. In terms of Impagliazzo's five world, SoI holds in Algorithmica, while SoI does not hold in Minicrypt. It has been a long-standing open question to determine whether SoI holds in Heuristica or Pessiland. Recently, building on [Hir21a], this open question was independently resolved by [Hir21b; GK22]: SoI holds in Heuristica.

Theorem 22 ([Hir21b; GK22])**.** *If* $\mathsf{DistNP} \subseteq \mathsf{AvgP}$*, then SoI holds.*

Proof. Let $\widetilde{\mathrm{K}}$ be the polynomial-time algorithm of Fact 7, which satisfies the property that there exists a polynomial p such that for every $x \in \{0,1\}^*$ and every $t \geq |x|$,

$$\mathrm{K}^{p(t)}(x) - \log p(t) \leq \widetilde{\mathrm{K}}(x; 1^t) \leq \mathrm{K}^t(x) \tag{5}$$

Fix strings $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$ and an integer $t \geq n + m$. The proof of SoI is given by analyzing the following three values for $z \sim \{0,1\}^{nk}$, $w \sim \{0,1\}^{nk+k}$, $z' \sim \{0,1\}^{mk}$, and $w' \sim \{0,1\}^{mk+k}$:

$$\begin{array}{lccc} \widetilde{\mathrm{K}}(& \mathrm{DP}_k(x;z), & \mathrm{DP}_\ell(y;z') & ; 1^{t'}), \\ \widetilde{\mathrm{K}}(& w, & \mathrm{DP}_\ell(y;z') & ; 1^{t'}), \\ \widetilde{\mathrm{K}}(& w, & w' & ; 1^{t'}), \end{array}$$

where $t' = t^{O(1)}$, $k \approx \mathrm{K}^t(x,y) - \ell$, and $\ell \approx \mathrm{K}^{\mathsf{poly}(t)}(y)$ are parameters chosen later.

First, observe that Fact 5 implies that $\mathrm{K}(w,w') \geq |w| + |w'| - 2$ with probability at least $\frac{3}{4}$. Let $\theta := |w| + |w'| - 2 - \log p(t')$; using Eq. (5), we obtain

$$\Pr_{w,w'}\left[\widetilde{\mathrm{K}}(w,w';1^{t'}) \geq \theta\right] \geq \frac{3}{4}. \tag{6}$$

Next, we set the parameter ℓ to be $\mathrm{K}^{p'(t)}(y) - \log p'(t) - 1$, where p' is some large polynomial. Consider a randomized circuit D that takes w' as input as well as random bits w and outputs 1 if and only if $\widetilde{\mathrm{K}}(w,w';1^{t'}) \geq \theta$. By the contrapositive of Lemma 12, $\mathrm{DP}_\ell(y;\text{-})$ is a pseudorandom generator secure against D; i.e., D cannot distinguish $\mathrm{DP}_\ell(y;z')$ and w' in the sense that

$$\left|\Pr_{w,z'}\left[\widetilde{\mathrm{K}}(w,\mathrm{DP}_\ell(y;z');1^{t'}) \geq \theta\right] - \Pr_{w,w'}\left[\widetilde{\mathrm{K}}(w,w';1^{t'}) \geq \theta\right]\right| < \frac{1}{4},$$

which, together with Eq. (6), implies that

$$\Pr_{w,z'}\left[\widetilde{\mathrm{K}}(w,\mathrm{DP}_\ell(y;z');1^{t'}) \geq \theta\right] \geq \frac{1}{2}.$$

Finally, we compare $\widetilde{\mathrm{K}}(w,\mathrm{DP}_\ell(y;z');1^{t'})$ with $\widetilde{\mathrm{K}}(\mathrm{DP}_k(x;z),\mathrm{DP}_\ell(y;z');1^{t'})$. On one hand, since $|w| = |z| + k$ and $|w'| = |z'| + \ell$, we have

$$\Pr_{w,z'}\left[\widetilde{\mathrm{K}}(w,\mathrm{DP}_\ell(y;z');1^{t'}) \geq |z| + |z'| + k + \ell - 2 - \log p(t')\right] \geq \frac{1}{2}. \tag{7}$$

On the other hand, observe that for some $t' := \mathsf{poly}(t)$,

$$\widetilde{\mathrm{K}}(\mathrm{DP}_k(x;z),\mathrm{DP}_\ell(y;z');1^{t'}) \leq \mathrm{K}^{t'}(\mathrm{DP}_k(x;z),\mathrm{DP}_\ell(y;z')) \leq \mathrm{K}^t(x,y) + |z| + |z'| + O(\log n)$$

holds because the strings $\mathrm{DP}_k(x;z)$ and $\mathrm{DP}_\ell(y;z')$ can be computed from k, ℓ, z, z', and a program of size $\mathrm{K}^t(x,y)$ that outputs (x,y) in time t. We now set $k := \mathrm{K}^t(x,y) - \ell + O(\log t)$ so that

$$\Pr_{z,z'}\left[\widetilde{\mathrm{K}}(\mathrm{DP}_k(x;z), \mathrm{DP}_\ell(y;z'); 1^{t'}) < |z| + |z'| + k + \ell - 2 - \log p(t')\right] = 1. \tag{8}$$

Let D_y be a randomized circuit that takes an input w and random bits z' and outputs 1 if and only if $\widetilde{\mathrm{K}}(w, \mathrm{DP}_\ell(y;z'); 1^{t'}) < |z| + |z'| + k + \ell - 2 - \log p(t')$. It follows from Eqs. (7) and (8) that

$$\Pr_{z,z'}\left[D_y(\mathrm{DP}_k(x;z); z') = 1\right] - \Pr_{w,z'}\left[D_y(w;z') = 1\right] \geq 1 - \frac{1}{2} = \frac{1}{2}.$$

Using Lemma 12, we obtain that

$$\mathrm{K}^{\mathsf{poly}(t)}(x \mid D_y) \leq k + O(\log t) = \mathrm{K}^t(x,y) - \mathrm{K}^{p'(t)}(y) + O(\log t).$$

It follows that for some large polynomial q,

$$\begin{aligned}\mathrm{K}^{q(t)}(x \mid y) \leq \mathrm{K}^t(x \mid D_y) + \mathrm{K}^t(D_y \mid y) + O(1) &\leq \mathrm{K}^t(x,y) - \mathrm{K}^{p'(t)}(y) + O(\log t)\\ &\leq \mathrm{K}^t(x,y) - \mathrm{K}^{q(t)}(y) + \log q(t)\end{aligned}$$

as desired. □

We mention that SoI holds if $\mathrm{Gap}(\mathrm{K}^{\mathrm{SAT}}$ vs K) is easy because $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ follows from $\mathrm{Gap}(\mathrm{K}^{\mathrm{SAT}}$ vs K) $\in \mathsf{P}$ [Hir20a].

Corollary 23. *If* $\mathrm{Gap}(\mathrm{K}^{\mathrm{SAT}}$ vs K) $\in \mathsf{P}$*, then SoI holds.*

We conclude this section by raising an open question.

Open Question 24. Does SoI hold in Pessiland? That is, does the non-existence of one-way functions imply SoI?

8 Universal Heuristic Scheme

Antunes and Fortnow [AF09] showed that the running time of a heuristic scheme that works with respect to every polynomial-time-samplable distribution can be characterized by using the notion of computational depth: Under a plausible assumption, $\{L\} \times \mathrm{PSAMP} \subseteq \mathsf{AvgP}$ if and only if there exists an algorithm S that decides L on input x in time $2^{O(\mathrm{cd}^{\mathsf{poly}(|x|)}(x) + \log|x|)}$, i.e., S runs in exponential time in the *time-unbounded* computational depth $\mathrm{cd}^{\mathsf{poly}(|x|),\infty}(x)$. Here, we study a faster algorithm that runs in exponential time in the *time-bounded* computational depth, which we call a universal heuristic scheme.

A universal heuristic scheme for L is an algorithm that takes an additional parameter t and decides L on input x in time $2^{O(\mathrm{cd}^{t,p(t)}(x) + \log t)}$. More formally:

Definition 25 (Universal Heuristic Scheme)**.** *An algorithm S is said to be a* universal heuristic scheme *for a language L if there exists a polynomial p such that for every x and every $t \geq p(|x|)$, the algorithm S outputs $L(x)$ on input (x,t) in time $2^{O(\mathrm{cd}^{t,p(t)}(x))} \cdot t^{O(1)}$.*

It is not hard to see that the existence of a universal heuristic scheme implies an algorithm that runs in time $2^{O(n/\log n)}$ on inputs of length n.

Theorem 26 ([Hir21a])**.** *If there exists a universal heuristic scheme S for a language L, then $L \in \mathsf{DTIME}\left(2^{O(n/\log n)}\right)$.*

Proof. Let p be the polynomial in Definition 25. We present an algorithm A that solves L on inputs of length n. Let x be an input of length n. Let I be a parameter chosen later, and let $k := 2n/I$. Define $t_i := p^{(i)}(n)$ for each $i \in [I]$. The algorithm A simulates the universal heuristic scheme S on inputs $(x, 1^{t_1}), (x, 1^{t_2}), \ldots, (x, 1^{t_I})$ in parallel. If one of the simulations halts with output being $b \in \{0, 1\}$, then A outputs b and halts. In other words, $A(x)$ is defined to be $S(x, 1^{t_i})$, where $i \in [I]$ is the index such that the running time of $S(x, 1^{t_i})$ is the smallest among $i \in [I]$.

We claim that A solves L in time $2^{O(n/\log n)}$. The correctness of A follows from the definition that S outputs the correct answer $L(x)$ when it halts. To bound the running time of A, consider the following telescoping sum:

$$\sum_{i=1}^{I} \mathrm{cd}^{t_i,t_{i+1}}(x) = \mathrm{cd}^{t_1,t_{I+1}}(x) \leq n + O(1),$$

where the last inequality follows because $\mathrm{K}^{t_1}(x) \leq n + O(1)$ and $\mathrm{K}^{t_{I+1}}(x) \geq 0$. By taking the minimum term of the left-hand side, we obtain

$$I \cdot \min\{\mathrm{cd}^{t_i,t_{i+1}}(x) \;\big|\; i \in [I]\} \leq n + O(1),$$

from which it follows that there exists $i \in [I]$ such that $\mathrm{cd}^{t_i,t_{i+1}}(x) \leq n/I + O(1) \leq k$. The running time of S on input $(x, 1^{t_i})$ is at most $2^{O\left(\mathrm{cd}^{t_i,t_{i+1}}(x)+\log t_i\right)}$. Let $c > 1$ be a constant such that $p(n) \leq n^c$ for all large n; then, we have $t_i = p^{(i)}(n) \leq n^{c^i}$ for all large n and every i. In particular, if we choose $I = \epsilon \log n$ for some small constant $\epsilon > 0$, we obtain $t_i \leq 2^{c^I \log n} \leq 2^{\sqrt{n}}$. We conclude that the running time of S on input $(x, 1^{t_i})$ is at most $2^{O\left(\mathrm{cd}^{t_i,t_{i+1}}(x)+\log t_i\right)} \leq 2^{O(n/\log n)}$. Thus, A also runs in time at most $2^{O(n/\log n)}$. □

Remark 27. Instead of the worst-case algorithm A, it is possible to construct an efficient heuristic algorithm using Theorem 9. Specifically, one can construct an algorithm A' such that given parameters $I \in \mathbb{N}$ and $\delta^{-1} \in \mathbb{N}$, with probability at least $1 - \delta$ over an instance x drawn from a distribution $\mathcal{D} \in \mathrm{PSAMP}$, the algorithm A' decides L on input x in time $2^{O\left((\log 1/\delta)/I + c^I \log n\right)}$ for some constant c.

In light of Theorem 26, in order to prove Theorem 2, it suffices to construct a universal heuristic scheme for each language in PH. It is easier to construct a weak variant of universal heuristic schemes, which we introduce below.

Definition 28 ([Hir21a])**.** *A* weak 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 $x \in \{0, 1\}^n$, if $\mathrm{cd}^{t,p(t)}(x) \leq k$, then $S(x, 1^t, 1^{2^k}) = L(x)$.*

We also introduce a strong variant that can check an input x is an easy instance or not in polynomial time.

Definition 29 ([Hir21b])**.** *A* strong 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.*

These different notions of universal heuristic schemes are in fact equivalent in Heuristica.

Theorem 30 ([Hir21a; Hir21b])**.** *Assume that* GapMINKT $\in \mathsf{P}$. *Then, the following are equivalent for any language L:*

1. *There exists a strong universal heuristic scheme for L.*
2. *There exists a universal heuristic scheme for L.*
3. *There exists a weak universal heuristic scheme for L.*

Moreover, under the stronger assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$*, the following statement is also equivalent.*

4. $\{L\} \times \mathrm{PSAMP} \subseteq \mathsf{Avg}_{\mathsf{P}}\mathsf{P}$.

Proof. It is easy to show Item 1 $\Rightarrow$ Item 2 $\Rightarrow$ Item 3. Let (S, C) be a strong universal heuristic scheme. We define a universal heuristic scheme S' as follows: Given $(x, 1^t)$ as input, S' finds the minimum $k \in \mathbb{N}$ such that $C(x, 1^t, 1^k) = 1$ and outputs $S(x, 1^t, 1^{2^k})$. Note that $k \leq \mathrm{cd}^{t,p(t)}(x)$ by the property of the string universal heuristic scheme (S, C); thus, the running time of S' is bounded by $2^{O(k+\log t)} \leq 2^{O(\mathrm{cd}^{t,p(t)}(x)+\log t)}$. Next, we show that any universal heuristic scheme S' can be converted into a weak universal heuristic scheme S''. We define S'' as follows: Given $(x, 1^t, 1^{2^k})$ as input, S'' simulates S' on input $(x, 1^t)$ up to $2^{O(k+\log t)}$ steps and outputs the output of S' if S' halts; if S' does not halt, the output is defined to be, e.g., 0. Then, S'' is a polynomial-time algorithm. The correctness of S'' follows from the definition that S' halts in time $2^{O(\mathrm{cd}^{t,p(t)}(x)+\log t)} \leq 2^{O(k+\log t)}$ if $\mathrm{cd}^{t,p(t)}(x) \leq k$.

We now show Item 3 $\Rightarrow$ Item 1. Given a weak universal heuristic scheme S for L, we need to construct a checker C. The idea of constructing C is to estimate the time-bounded computational depth of an input by using the polynomial-time algorithm for GapMINKT whose existence is guaranteed by Theorem 11. Let $\widetilde{\mathrm{K}}$ be the polynomial-time algorithm of Fact 7 such that for every $x \in \{0,1\}^*$ and every $t \geq |x|$,[11]

$$\mathrm{K}^{p(t)}(x) - \log p(t) \leq \widetilde{\mathrm{K}}(x, 1^t) \leq \mathrm{K}^t(x).$$

Observe that

$$\mathrm{cd}^{p(t),p^{(2)}(t)}(x) - \log p(t) \leq \widetilde{\mathrm{K}}(x, 1^t) - \widetilde{\mathrm{K}}(x, 1^{p^{(2)}(t)}) \leq \mathrm{cd}^{t,p^{(3)}(t)}(x) + \log p^{(3)}(t). \tag{9}$$

We define a checker C as follows: $C(x, 1^t, 1^k) = 1$ if and only if $\widetilde{\mathrm{K}}(x, 1^t) - \widetilde{\mathrm{K}}(x, 1^{p^{(2)}(t)}) \leq k + \log p^{(3)}(t)$. We define a solver S' so that $S'(x, 1^t, 1^{2^k}) := S'(x, 1^{p(t)}, 1^{2^{k'}})$, where $k' := k + \log p(t) + \log p^{(3)}(t)$.

Below, we claim that (S', C) is a strong universal heuristic scheme by showing that it satisfies the two properties of Definition 29.

[11] We may assume without loss of generality that the polynomial p in Fact 7 is the same polynomial with Definition 28.

1. If $\mathrm{cd}^{t,p^{(3)}(t)}(x) \le k$, then by the upper bound of Eq. (9), we have $C(x, 1^t, 1^k) = 1$.

2. If $C(x, 1^t, 1^k) = 1$, then by the definition of C and by the lower bound of Eq. (9), we obtain $\mathrm{cd}^{p(t),p^{(2)}(t)}(x) \le k + \log p(t) + \log p^{(3)}(t) = k'$. It follows from the property of the weak heuristic scheme S that $S'(x, 1^t, 1^{2^k}) = S(x, 1^{p(t)}, 1^{2^{k'}}) = L(x)$.

The implication from Item 1 to 4 can be proved using Theorem 9. The converse can be proved by considering the "time-bounded universal distribution". We omit the detailed proof of the equivalence between Item 1 and Item 4, which can be found in [Hir21a]. □

9 Constructing Universal Heuristic Schemes

We now use SoI to construct a weak universal heuristic scheme for every language in PH. For simplicity, we first construct a weak universal heuristic for every language in NP.

Theorem 31. *If* $\mathrm{Gap}(\mathrm{K}^{\mathrm{SAT}}\text{ vs }\mathrm{K}) \in \mathsf{P}$*, then for every language* $L \in \mathsf{NP}$*, there exists a weak universal heuristic scheme for* L*.*

Proof. Let V be a polynomial-time verifier for $L \in \mathsf{NP}$. For every $x \in L$, let y_x denote the lexicographically first certificate y_x such that $V(x, y_x) = 1$. The following claim is the key to the construction of a weak universal heuristic scheme.

Claim 32. *There exists a polynomial* q *such that for every* $x \in L$ *and every* $t \ge |x|$*,*

$$\mathrm{K}^{q(t)}(y_x \mid x) \le \mathrm{cd}^{t,q(t)}(x) + \log q(t).$$

Proof. Note that SoI holds because of Corollary 23. Using SoI, we obtain

$$\begin{aligned}
&\mathrm{K}^{p^{(3)}(t)}(y_x \mid x) \\
&\le \mathrm{K}^{p^{(2)}(t)}(y_x, x) - \mathrm{K}^{p^{(3)}(t)}(x) + \log p^{(3)}(t) && \text{(by SoI)} \\
&\le \mathrm{K}^{p(t),\mathrm{SAT}}(y_x, x) + \log p^{(2)}(t) - \mathrm{K}^{p^{(3)}(t)}(x) + \log p^{(3)}(t) && \text{(by Lemma 20)} \\
&\le \mathrm{K}^{t}(x) - \mathrm{K}^{p^{(3)}(t)}(x) + O(\log p^{(3)}(t)) \\
&= \mathrm{cd}^{t,p^{(3)}(t)}(x) + O(\log p^{(3)}(t)),
\end{aligned}$$

where the last inequality holds because y_x can be computed from x in polynomial time given oracle access to SAT. The claim follows by letting $q(t) := p^{(3)}(t)^{O(1)}$. ◇

We now present a weak universal heuristic scheme S for L: The algorithm S takes $(x, 1^t, 1^{2^k})$ as input and computes the set Y of strings $y \in \{0,1\}^*$ such that there exists a program of length at most $k + \log q(t)$ that takes x as input and outputs y in time $q(t)$. In other words, we define

$$Y := \left\{ y \in \{0,1\}^* \;\middle|\; \mathrm{K}^{q(t)}(y \mid x) \le k + \log q(t). \right\}$$

Note that $|Y| \le 2^{k+\log q(t)+1}$ and Y can be computed in time $\mathsf{poly}(|x|, t, 2^k)$ by enumerating all the programs of length at most $k + \log q(t)$. The algorithm S outputs 1 if and only if there exists a string $y \in Y$ such that $V(x, y) = 1$. Clearly, S is a polynomial-time algorithm. We prove the correctness of S: It is evident that S does not err on any input $x \notin L$. Consider an input $x \in L$ such that $\mathrm{cd}^{t,q(t)}(x) \le k$. Then, by Claim 32 we have $\mathrm{K}^{q(t)}(y_x) \le k + \log q(t)$, which implies $y_x \in Y$ and thus S accepts. □

This enables us to complete a proof of a special case of Theorem 2.

Corollary 33. *If* $\mathsf{DistPH} \subseteq \mathsf{AvgP}$*, then* $\mathsf{NP} \subseteq \mathsf{DTIME}(2^{O(n/\log n)})$.

Proof. See Fig. 4. □

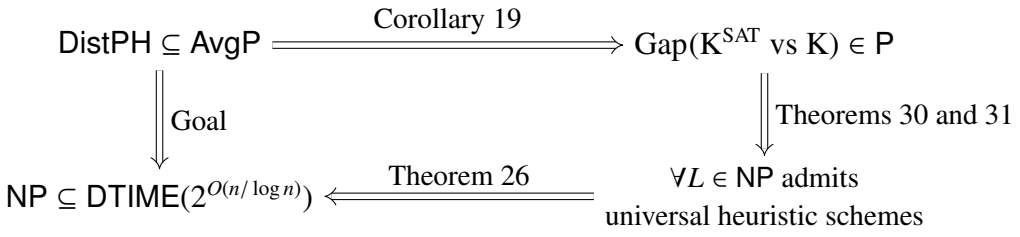

Figure 4: The proof of Corollary 33.

To extend Theorem 31 to all the levels of PH, we use an inductive argument that constructs weak universal heuristic schemes for the k-th level Σ_k^{p} of PH from weak universal heuristic schemes for $\Sigma_{k-1}^{\mathsf{p}}$.

Theorem 34. *Let* $k \in \mathbb{N}$*. If* $\mathsf{Dist}\Sigma_{k+1}^{\mathsf{p}} \subseteq \mathsf{AvgP}$*, then for every language* $L \in \Sigma_k^{\mathsf{p}}$*, there exists a weak universal heuristic scheme for* L*.*

Proof. We prove this by induction on $k \in \mathbb{N}$. The base case ($k = 0$) is trivial because every language $L \in \Sigma_0^{\mathsf{p}} = \mathsf{P}$ admits a weak universal heuristic scheme. Let $k \geq 1$. Let V be a language in Π_{k-1}^{p} such that $x \in L$ if and only if $V(x, y) = 1$ for some $y \in \{0,1\}^{\mathsf{poly}(|x|)}$. For every $x \in L$, let y_x be the lexicographically first string y such that $V(x, y) = 1$. Using the same proof idea of Claim 32, it is easy to prove the following.

Claim 35. *There exists a polynomial* q *such that for every* $x \in L$ *and every* $t \geq |x|$*,*

$$\mathrm{K}^{q(t)}(y_x \mid x) \leq \mathrm{cd}^{t,q(t)}(x) + \log q(t).$$

By the induction hypothesis, there exists a weak universal heuristic scheme S for $V \in \Pi_{k-1}^{\mathsf{p}}$. Let p be the polynomial in Definition 28. Using S, we now present a weak universal heuristic scheme S' for L: The algorithm S' takes $(x, 1^t, 1^{2^k})$ as input and computes the set

$$Y := \left\{ y \in \{0,1\}^* \;\middle|\; \mathrm{K}^{q(t)}(y \mid x) \leq k + \log q(t). \right\}$$

Note that $|Y| \leq 2^{k+\log q(t)+1}$ and Y can be computed in time $\mathsf{poly}(|x|, t, 2^k)$ by an exhaustive search. The algorithm S' outputs 1 if and only if there exists a string $y \in Y$ such that $S((x, y), 1^{t'}, 1^{2^{k'}}) = 1$, where $t' = t^{O(1)}$ and $k' = O(k + \log t)$ are parameters chosen later. Clearly, S' is a polynomial-time algorithm.

We claim the correctness of S'. Assume that $\mathrm{cd}^{t,2p(t')}(x) \leq k$. We claim that for some parameter $t' = q(t)^{O(1)}$ and for every $y \in Y$, the $(t', p(t'))$-time-bounded computational depth of (x, y) is at most k', which will imply that the output of the weak universal heuristic scheme S is correct on input (x, y). For every $y \in Y$, we have

$$\begin{aligned}
\mathrm{cd}^{t',p(t')}(x, y) &\leq \mathrm{K}^{q(t)}(x) + \mathrm{K}^{q(t)}(y \mid x) - \mathrm{K}^{p(t')}(x, y) + O(1) \\
&\leq \mathrm{cd}^{q(t),2p(t')}(x) + k + \log q(t) + O(1) \\
&\leq 2k + \log q(t) + O(1) =: k',
\end{aligned}$$

where the first inequality follows from the definition of time-bounded computational depth, the second inequality follows from the fact that $\mathrm{K}^{2p(t')}(x) \leq \mathrm{K}^{p(t')}(x,y) + O(1)$ and $y \in Y$, and the third inequality follows from the assumption that $\mathrm{cd}^{q(t),2p(t')}(x) \leq \mathrm{cd}^{t,q'(t)}(x) \leq k$. By the correctness of the weak universal heuristic scheme S, we obtain $S((x,y), 1^{t'}, 1^{2^{k'}}) = V(x,y)$. If $x \in L$, Claim 35 implies that $y_x \in Y$; thus, we have $S((x,y_x), 1^{t'}, 1^{2^{k'}}) = V(x,y_x) = 1$, which implies that S' outputs 1. If $x \notin L$, then $V(x,y) = 0$ for every string y; thus, we obtain $S((x,y), 1^{t'}, 1^{2^{k'}}) = V(x,y) = 0$, which implies that S' outputs 0. □

This completes the proof of the second item of Theorem 2, as shown in Fig. 5.

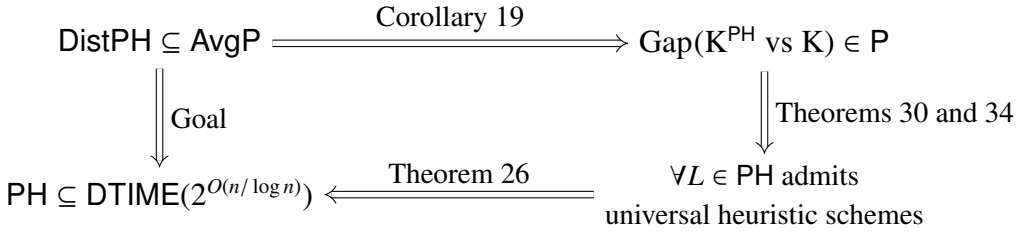

Figure 5: The proof of the second item of Theorem 2.

Finally, we construct universal heuristic schemes for UP.

Theorem 36. *If* DistNP ⊆ AvgP*, then for every language* $L \in$ UP*, there exists a weak universal heuristic scheme for* L*.*

Proof. Let V be a UP-type verifier for L; that is, for every $x \in L$, there exists a unique certificate y_x such that $V(x,y_x) = 1$. Consider an ensemble $L = \{L_{\langle n,t,s\rangle}\}_{n,t,s\in\mathbb{N}}$ of languages defined as

$$L_{\langle n,t,s\rangle} := \{(x,y) \mid x \in \{0,1\}^n, V(x,y) = 1, \mathrm{K}^t(x) \leq s\}.$$

It is easy to observe that $L \in$ NP. Using that $L \in$ UP, we can observe that $|L_{\langle n,t,s\rangle}| \leq 2^{s+1}$ because the number of strings x such that $\mathrm{K}^t(x) \leq s$ is at most 2^{s+1} by Fact 5 and for each x there is at most one certificate y such that $V(x,y) = 1$. Applying Theorem 16 to L, there exists a polynomial p such that for every $t \geq n$, for every $(x,y_x) \in L_{\langle n,t,s\rangle}$ such that $s := \mathrm{K}^t(x)$, it holds that

$$\mathrm{K}^{p(t)}(x,y_x) \leq s + 1 + \log p(t) = \mathrm{K}^t(x) + \log 2p(t).$$

Since SoI follows from Theorem 22, we have

$$\mathrm{K}^{p^{(2)}(t)}(y_x \mid x) \leq \mathrm{K}^{p(t)}(x,y_x) - \mathrm{K}^{p^{(2)}(t)}(x) + \log p^{(2)}(t).$$

Combining these two inequalities, we obtain

$$\mathrm{K}^{p^{(2)}(t)}(y_x \mid x) \leq \mathrm{cd}^{t,p^{(2)}(t)}(x) + O(\log t). \tag{10}$$

We now present a weak universal heuristic scheme S for L. Given an input $(x, 1^t, 1^{2^k})$ such that $\mathrm{cd}^{t,p^{(2)}(t)}(x) \leq k$, the algorithm S computes the set

$$Y := \left\{y \in \{0,1\}^* \;\middle|\; \mathrm{K}^{p^{(2)}(t)}(y \mid x) \leq k + O(\log t)\right\}$$

and accepts if and only if there exists $y \in Y$ such that $V(x,y) = 1$. The correctness of S follows from Eq. (10) because it implies $y_x \in Y$ for every $x \in L$. □

This enables us to complete the proof of Theorem 2.

Proof of the first item of Theorem 2. Assume $\mathsf{DistNP} \subseteq \mathsf{AvgP}$. By Theorem 36, every language $L \in \mathsf{UP}$ admits a *weak* universal heuristic scheme, which can be converted into a universal heuristic scheme by Theorem 30, from which we obtain $L \in \mathsf{DTIME}(2^{O(n/\log n)})$ by Theorem 26. □

In the original paper [Hir21a], a more general result than Theorem 36 was proved: $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ implies that $\mathsf{NP}_{\mathsf{sv}}$ admits universal heuristic schemes. Here, $\mathsf{NP}_{\mathsf{sv}}$ stands for *size-verifiable* NP and is the class of languages $L \in \mathsf{NP}$ such that some AM protocols can verify that the number of certificates for L is approximately small. The notion of size-verifiability was introduced by Akavia, Goldreich, Goldwasser, and Moshkovitz [AGGM06]. It is easy to observe that $\mathsf{UP} \subseteq \mathsf{FewP} \subseteq \mathsf{NP}_{\mathsf{sv}} \subseteq \mathsf{NP}$ and that $\mathsf{NP}_{\mathsf{sv}} = \mathsf{NP}$ if $\mathsf{NP} \subseteq \mathsf{coAM}$; however, the complexity of $\mathsf{NP}_{\mathsf{sv}}$ is not well understood. It is an interesting open problem to extend UP of Theorem 36 to NP.

Open Question 37. Does $\mathsf{NP} \nsubseteq \mathsf{DTIME}(2^{O(n/\log n)})$ imply $\mathsf{DistNP} \nsubseteq \mathsf{AvgP}$? In particular, is it possible to construct universal heuristic schemes for NP under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$?

10 Future Research Directions

We conclude this article by presenting three research directions which we believe are most promising and exciting.

The first research direction is to develop non-relativizing proof techniques. The only non-relativizing part of the proofs in this article is Theorem 3, which constructs a complexity-theoretic pseudorandom generator in Heuristica. In fact, there is a relativizing proof showing the existence of a pseudorandom generator from the stronger assumption that $\mathsf{DistP}^{\mathsf{NP}} \subseteq \mathsf{AvgP}$ [HN21]. Given that there is a quantitatively tight relativization barrier [HN21], we would need a non-relativizing proof technique to obtain better worst-case-to-average-case connections for NP or PH. A recent line of work [AHMPS08; HOS18; Ila20a; ILO20; Ila20b; ACMTV21; LP21; Hir21b] developed apparently non-relativizing proof techniques: Although Ko [Ko91] showed that GapMINKT cannot be shown to be NP-hard using a relativizing proof technique, NP-hardness of computing sublinear-time-bounded conditional Kolmogorov complexity is proved in [ACMTV21; LP21; Hir21b]. Note that NP-hardness of GapMINKT excludes Heuristica by Theorem 11; even NP-hardness of $\mathrm{GapMINKT}^{\mathsf{PH}}$ would significantly improve Theorem 2. Trying to prove NP-hardness of meta-computational problems would lead us to new non-relativizing proof techniques. A state-of-the-art result along this research line is due to Ilango [Ila20b; Ila21], who showed NP-hardness of variants of the Minimum Circuit Size Problem [KC00], which is another representative meta-computational problem.

The second research direction is to use meta-complexity to exclude Pessiland. Impagliazzo and Levin [IL90, Proposition 1] presented a proof sketch of the characterization of the existence of a one-way function: there exists no one-way function if and only if the randomized t-time-bounded Kolmogorov complexity of x can be approximated with high probability over a random input x drawn from any unknown t-time-samplable distribution. Their results can be seen as an approach toward excluding Pessiland: If the randomized t-time-bounded Kolmogorov complexity is NP-hard under t'-time randomized reductions for $t' \ll t$, then Pessiland does

not exist, i.e, (error-prone) average-case hardness of NP implies the existence of a one-way function. More recently, Liu and Pass [LP20] proved the equivalence between the non-existence of a one-way function and the existence of a randomized polynomial-time *error-prone* heuristic that computes $\mathrm{K}^t(x)$ with high probability over a random input x chosen from the uniform distribution. The main gap between these results and the results presented in this article is the difference between *error-prone* average-case complexity (denoted by HeurP in [BT06a]) and *errorless* average-case complexity (AvgP), which was recently investigated in [HS22a]. An intermediate statement is SoI, which is sandwiched between errorless heuristics for MINKT and error-prone heuristics for MINKT [Hir21b]. Nanashima [Nan21] showed that the existence of a one-way function follows from $\mathsf{NP} \not\subseteq \mathsf{BPP}$ (i.e., both Heuristica and Pessiland can be excluded) if there exists a randomized nonadaptive reduction from NP to the adversary of auxiliary-input hitting set generators. Note that the existence of a hitting set generator is equivalent to the hardness of GapMINKT, at least under non-black-box reductions (Proposition 13). Given these results, we conjecture that meta-complexity would play a central role in excluding Pessiland, as well as Heuristica.

The last research direction is to determine average-case complexity of natural distributional problems in DistNP. We started off this article by explaining that the motivation of studying average-case complexity is to understand the "real-life" complexity of (natural) distributional problems. The original motivation of Levin [Lev86], who laid the foundation of average-case complexity, was also the same and showed that the Tiling problem is DistNP-complete. However, a relatively few distributional problems in DistNP are shown to be DistNP-complete, compared to the highly successful theory of NP-completeness [Coo71; Lev73; Kar72]. The difficulty is that an average-case reduction disturbs the distribution of distributional problems, which makes it difficult to prove DistNP-completeness of distributional problems $(L, \mathcal{D})$ for natural distributions $\mathcal{D}$. We envision that meta-complexity could come to the rescue: Meta-complexity enables us to analyze *average-case* complexity through the lens of *worst-case* complexity, for which a reduction is easier to construct. [Hir20a; HS22b] showed that $(\mathrm{MINKT}^{\mathsf{PH}}, \mathcal{U})$ is DistPH-complete, which may be a step toward determining average-case complexity of natural distributional problems in DistPH or DistNP.

References

[ACMTV21] Eric Allender, Mahdi Cheraghchi, Dimitrios Myrisiotis, Harsha Tirumala, and Ilya Volkovich. "One-Way Functions and a Conditional Variant of MKTP". In: *Proceedings of the Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*. 2021, 7:1–7:19. DOI: 10.4230/LIPIcs.FSTTCS.2021.7.

[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)*. 2009, pp. 298–303. DOI: 10.1109/CCC.2009.12.

[AFMV06] Luis Antunes, Lance Fortnow, Dieter van Melkebeek, and N. V. Vinodchandran. "Computational depth: Concept and applications". In: *Theor. Comput. Sci.* 354.3 (2006), pp. 391–404. DOI: 10.1016/j.tcs.2005.11.033.

[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)*. 2006, pp. 701–710. DOI: 10.1145/1132516.1132614.

[AHMPS08] 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". In: *SIAM J. Comput.* 38.1 (2008), pp. 63–84. DOI: 10.1137/060664537.

[All21] Eric Allender. "Vaughan Jones, Kolmogorov Complexity, and the new complexity landscape around circuit minimization". In: *New Zealand journal of mathematics* 52 (2021), pp. 585–604.

[AW09] Scott Aaronson and Avi Wigderson. "Algebrization: A New Barrier in Complexity Theory". In: *TOCT* 1.1 (2009), 2:1–2:54. DOI: 10.1145/1490270.1490272.

[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)*. 2015, pp. 1–6. DOI: 10.1007/978-3-662-46494-6_1.

[BCGL92] Shai Ben-David, Benny Chor, Oded Goldreich, and Michael Luby. "On the Theory of Average Case Complexity". In: *J. Comput. Syst. Sci.* 44.2 (1992), pp. 193–219. DOI: 10.1016/0022-0000(92)90019-F.

[BFL01] Harry Buhrman, Lance Fortnow, and Sophie Laplante. "Resource-Bounded Kolmogorov Complexity Revisited". In: *SIAM J. Comput.* 31.3 (2001), pp. 887–905. DOI: 10.1137/S009753979834388X.

[BFP05] Harry Buhrman, Lance Fortnow, and Aduri Pavan. "Some Results on Derandomization". In: *Theory Comput. Syst.* 38.2 (2005), pp. 211–227. DOI: 10.1007/s00224-004-1194-y.

[BGS75] Theodore P. Baker, John Gill, and Robert Solovay. "Relativizations of the P =? NP Question". In: *SIAM J. Comput.* 4.4 (1975), pp. 431–442. DOI: 10.1137/0204037.

[BLM05] Harry Buhrman, Troy Lee, and Dieter van Melkebeek. "Language compression and pseudorandom generators". In: *Computational Complexity* 14.3 (2005), pp. 228–255. DOI: 10.1007/s00037-005-0199-5.

[BRSV17] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. "Average-case fine-grained hardness". In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 2017, pp. 483–496. DOI: 10.1145/3055399.3055466.

[BRSV18] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. "Proofs of Work From Worst-Case Assumptions". In: *Proceedings of the International Cryptology Conference (CRYPTO)*. 2018, pp. 789–819. DOI: 10.1007/978-3-319-96884-1_26.

[BT06a] Andrej Bogdanov and Luca Trevisan. "Average-Case Complexity". In: *Foundations and Trends in Theoretical Computer Science* 2.1 (2006). DOI: 10.1561/0400000004.

[BT06b] Andrej Bogdanov and Luca Trevisan. “On Worst-Case to Average-Case Reductions for NP Problems”. In: *SIAM J. Comput.* 36.4 (2006), pp. 1119–1159. DOI: 10.1137/S0097539705446974.

[CHOPRS20] Lijie Chen, Shuichi Hirahara, Igor Carboni Oliveira, Ján Pich, Ninad Rajgopal, and Rahul Santhanam. “Beyond Natural Proofs: Hardness Magnification and Locality”. In: *Proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*. 2020, 70:1–70:48. DOI: 10.4230/LIPIcs.ITCS.2020.70.

[CHV22] Lijie Chen, Shuichi Hirahara, and Neekon Vafa. “Average-case Hardness of NP and PH from Worst-case Fine-grained Assumptions”. In: *Proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*. 2022, 67:1–67:17.

[Coo71] Stephen A. Cook. “The Complexity of Theorem-Proving Procedures”. In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 1971, pp. 151–158. DOI: 10.1145/800157.805047.

[DN92] Cynthia Dwork and Moni Naor. “Pricing via Processing or Combatting Junk Mail”. In: *Proceedings of the International Cryptology Conference (CRYPTO)*. 1992, pp. 139–147. DOI: 10.1007/3-540-48071-4_10.

[FF93] Joan Feigenbaum and Lance Fortnow. “Random-Self-Reducibility of Complete Sets”. In: *SIAM J. Comput.* 22.5 (1993), pp. 994–1005. DOI: 10.1137/0222061.

[FFLS94] Joan Feigenbaum, Lance Fortnow, Carsten Lund, and Daniel A. Spielman. “The Power of Adaptiveness and Additional Queries in Random-Self-Reductions”. In: *Comput. Complex.* 4 (1994), pp. 158–174. DOI: 10.1007/BF01202287.

[GK22] Halley Goldberg and Valentine Kabanets. “A Simpler Proof of the Worst-Case to Average-Case Reduction for Polynomial Hierarchy via Symmetry of Information”. In: *Electronic Colloquium on Computational Complexity (ECCC)* 007 (2022).

[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)*. 1989, pp. 25–32. DOI: 10.1145/73007.73010.

[Gol06] Oded Goldreich. “On Promise Problems: A Survey”. In: *Theoretical Computer Science, Essays in Memory of Shimon Even*. 2006, pp. 254–290. DOI: 10.1007/11685654_12.

[GS87] Yuri Gurevich and Saharon Shelah. “Expected Computation Time for Hamiltonian Path Problem”. In: *SIAM J. Comput.* 16.3 (1987), pp. 486–502. DOI: 10.1137/0216034.

[GVW11] Oded Goldreich, Salil P. Vadhan, and Avi Wigderson. “Simplified Derandomization of BPP Using a Hitting Set Generator”. In: *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*. Springer, 2011, pp. 59–67. DOI: 10.1007/978-3-642-22670-0_8.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. “A Pseudorandom Generator from any One-way Function”. In: *SIAM J. Comput.* 28.4 (1999), pp. 1364–1396. DOI: 10.1137/S0097539793244708.

[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)*. 2018, pp. 247–258.

[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)*. 2020, pp. 50–60.

[Hir20b] Shuichi Hirahara. “Non-Disjoint Promise Problems from Meta-Computational View of Pseudorandom Generator Constructions”. In: *Proceedings of the Computational Complexity Conference (CCC)*. 2020, 20:1–20:47. DOI: 10.4230/LIPIcs.CCC.2020.20.

[Hir20c] Shuichi Hirahara. “Unexpected hardness results for Kolmogorov complexity under uniform reductions”. In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 2020, pp. 1038–1051. DOI: 10.1145/3357713.3384251.

[Hir20d] Shuichi Hirahara. “Unexpected Power of Random Strings”. In: *Proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*. 2020, 41:1–41:13. DOI: 10.4230/LIPIcs.ITCS.2020.41.

[Hir21a] Shuichi Hirahara. “Average-case hardness of NP from exponential worst-case hardness assumptions”. In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 2021, pp. 292–302. DOI: 10.1145/3406325.3451065.

[Hir21b] Shuichi Hirahara. “Symmetry of Information in Heuristica”. manuscript. 2021.

[HN21] Shuichi Hirahara and Mikito Nanashima. “On Worst-Case Learning in Relativized Heuristica”. In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 2021.

[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)*. 2018, 5:1–5:31. DOI: 10.4230/LIPIcs.CCC.2018.5.

[HS21] Shuichi Hirahara and Nobutaka Shimizu. “Nearly Optimal Average-Case Complexity of Counting Bicliques Under SETH”. In: *Proceedings of the Symposium on Discrete Algorithms (SODA)*. 2021, pp. 2346–2365. DOI: 10.1137/1.9781611976465.140.

[HS22a] Shuichi Hirahara and Rahul Santhanam. “Errorless versus Error-prone Average-case Complexity”. In: *Proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*. 2022, 38:1–38:23.

[HS22b] Shuichi Hirahara and Rahul Santhanam. “Excluding PH Pessiland”. In: *Proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*. 2022, 78:1–78:25.

[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)*. 2020, 15:1–15:14. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2020.15.

[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)*. 1989, pp. 230–235. DOI: 10.1109/SFCS.1989.63483.

[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)*. 1990, pp. 812–821. DOI: 10.1109/FSCS.1990.89604.

[Ila20a] Rahul Ilango. “Approaching MCSP from Above and Below: Hardness for a Conditional Variant and $\mathsf{AC}^0[p]$”. In: *Proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*. 2020, 34:1–34:26. DOI: 10.4230/LIPIcs.ITCS.2020.34.

[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)*. 2020, pp. 424–433.

[Ila21] Rahul Ilango. “The Minimum Formula Size Problem is (ETH) Hard”. In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 2021.

[ILO20] Rahul Ilango, Bruno Loff, and Igor Carboni Oliveira. “NP-Hardness of Circuit Minimization for Multi-Output Functions”. In: *Proceedings of the Computational Complexity Conference (CCC)*. 2020, 22:1–22:36. DOI: 10.4230/LIPIcs.CCC.2020.22.

[Imp11] Russell Impagliazzo. “Relativized Separations of Worst-Case and Average-Case Complexities for NP”. In: *Proceedings of the Conference on Computational Complexity (CCC)*. 2011, pp. 104–114. DOI: 10.1109/CCC.2011.34.

[Imp95] Russell Impagliazzo. “A Personal View of Average-Case Complexity”. In: *Proceedings of the Structure in Complexity Theory Conference*. 1995, pp. 134–147. DOI: 10.1109/SCT.1995.514853.

[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)*. 1997, pp. 220–229. DOI: 10.1145/258533.258590.

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

[KC00] Valentine Kabanets and Jin-yi Cai. “Circuit minimization problem”. In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 2000, pp. 73–79. DOI: 10.1145/335305.335314.

[Ko91] Ker-I Ko. “On the Complexity of Learning Minimum Time-Bounded Turing Machines”. In: *SIAM J. Comput.* 20.5 (1991), pp. 962–986. DOI: 10.1137/0220059.

[KS04] Johannes Köbler and Rainer Schuler. “Average-case intractability vs. worst-case intractability”. In: *Inf. Comput.* 190.1 (2004), pp. 1–17. DOI: 10.1016/j.ic.2003.05.002.

[Lev73] Leonid Anatolevich Levin. “Universal sequential search problems”. In: *Problemy Peredachi Informatsii* 9.3 (1973), pp. 115–116.

[Lev84] Leonid A. Levin. “Randomness Conservation Inequalities; Information and Independence in Mathematical Theories”. In: *Information and Control* 61.1 (1984), pp. 15–37. DOI: 10.1016/S0019-9958(84)80060-1.

[Lev86] Leonid A. Levin. “Average Case Complete Problems”. In: *SIAM J. Comput.* 15.1 (1986), pp. 285–286. DOI: 10.1137/0215020.

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

[LP21] Yanyi Liu and Rafael Pass. “On One-way Functions from NP-Complete Problems”. In: *Electron. Colloquium Comput. Complex.* 28 (2021), p. 59.

[LV19] Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications, 4th Edition*. Texts in Computer Science. Springer, 2019. ISBN: 978-3-030-11297-4. DOI: 10.1007/978-3-030-11298-1.

[LW95] Luc Longpré and Osamu Watanabe. “On Symmetry of Information and Polynomial Time Invertibility”. In: *Inf. Comput.* 121.1 (1995), pp. 14–22. DOI: 10.1006/inco.1995.1120.

[MR07] Daniele Micciancio and Oded Regev. “Worst-Case to Average-Case Reductions Based on Gaussian Measures”. In: *SIAM J. Comput.* 37.1 (2007), pp. 267–302. DOI: 10.1137/S0097539705447360.

[Nan21] Mikito Nanashima. “On Basing Auxiliary-Input Cryptography on NP-Hardness via Nonadaptive Black-Box Reductions”. In: *Proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*. 2021, 29:1–29:15. DOI: 10.4230/LIPIcs.ITCS.2021.29.

[NW94] Noam Nisan and Avi Wigderson. “Hardness vs Randomness”. In: *J. Comput. Syst. Sci.* 49.2 (1994), pp. 149–167. DOI: 10.1016/S0022-0000(05)80043-1.

[RR97] Alexander A. Razborov and Steven Rudich. “Natural Proofs”. In: *J. Comput. Syst. Sci.* 55.1 (1997), pp. 24–35. DOI: 10.1006/jcss.1997.1494.

[Rud97] Steven Rudich. “Super-bits, Demi-bits, and NP/qpoly-natural Proofs”. In: *Proceedings of the Randomization and Approximation Techniques in Computer Science (RANDOM/APPROX)*. 1997, pp. 85–93. DOI: 10.1007/3-540-63248-4_8.

[Sho99] Peter W. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM Review* 41.2 (1999), pp. 303–332. DOI: 10.1137/S0036144598347011.

[Sip83] Michael Sipser. “A Complexity Theoretic Approach to Randomness”. In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 1983, pp. 330–335. DOI: 10.1145/800061.808762.

[Tre01] Luca Trevisan. “Extractors and pseudorandom generators”. In: *J. ACM* 48.4 (2001), pp. 860–879. DOI: 10.1145/502090.502099.

[TUZ07] Amnon Ta-Shma, Christopher Umans, and David Zuckerman. “Lossless Condensers, Unbalanced Expanders, And Extractors”. In: *Combinatorica* 27.2 (2007), pp. 213–240. DOI: 10.1007/s00493-007-0053-2.

[TV07] Luca Trevisan and Salil P. Vadhan. “Pseudorandomness and Average-Case Complexity Via Uniform Reductions”. In: *Computational Complexity* 16.4 (2007), pp. 331–364. DOI: 10.1007/s00037-007-0233-x.

[Uma09] Christopher Umans. “Reconstructive Dispersers and Hitting Set Generators”. In: *Algorithmica* 55.1 (2009), pp. 134–156. DOI: 10.1007/s00453-008-9266-z.

[Vad12] Salil P. Vadhan. “Pseudorandomness”. In: *Foundations and Trends in Theoretical Computer Science* 7.1-3 (2012), pp. 1–336. DOI: 10.1561/0400000010.

[Vio05a] Emanuele Viola. “On Constructing Parallel Pseudorandom Generators from One-Way Functions”. In: *Proceedings of the Conference on Computational Complexity (CCC)*. 2005, pp. 183–197. DOI: 10.1109/CCC.2005.16.

[Vio05b] Emanuele Viola. “The complexity of constructing pseudorandom generators from hard functions”. In: *Computational Complexity* 13.3-4 (2005), pp. 147–188. DOI: 10.1007/s00037-004-0187-1.

[Wat12] Thomas Watson. “Relativized Worlds without Worst-Case to Average-Case Reductions for NP”. In: *TOCT* 4.3 (2012), 8:1–8:30. DOI: 10.1145/2355580.2355583.

[Yap83] Chee-Keng Yap. “Some Consequences of Non-Uniform Conditions on Uniform Classes”. In: *Theor. Comput. Sci.* 26 (1983), pp. 287–300. DOI: 10.1016/0304-3975(83)90020-8.

[ZL70] AK Zvonkin and LA Levin. “The complexity of finite objects and the algorithmic concepts of randomness and information”. In: *UMN (Russian Math. Surveys)* 25.6 (1970), pp. 83–124.