

Excluding PH Pessiland

Shuichi Hirahara 

Principle of Informatics Research Division, National Institute of Informatics, Tokyo, Japan

Rahul Santhanam 

Department of Computer Science, University of Oxford, UK

Abstract

Heuristica and Pessiland are “worlds” of average-case complexity [Impagliazzo95] that are considered unlikely but that current techniques are unable to rule out. Recently, [Hirahara20] considered a PH (Polynomial Hierarchy) analogue of Heuristica, and showed that to rule it out, it would be sufficient to prove the NP-completeness of the problem GapMINKT^{PH} of estimating the PH-oracle time-bounded Kolmogorov complexity of a string.

In this work, we analogously define “PH Pessiland” to be a world where PH is hard on average but PH-computable pseudo-random generators do not exist. We unconditionally rule out PH-Pessiland in both non-uniform and uniform settings, by showing that the distributional problem of computing PH-oracle time-bounded Kolmogorov complexity of a string over the uniform distribution is complete for an (error-prone) average-case analogue of PH. Moreover, we show the equivalence between error-prone average-case hardness of PH and the existence of PH-computable pseudorandom generators.

2012 ACM Subject Classification Theory of computation → Complexity classes; Theory of computation → Problems, reductions and completeness

Keywords and phrases average-case complexity, pseudorandomness, meta-complexity

Digital Object Identifier 10.4230/LIPIcs.ITCS.2022.85

Funding *Shuichi Hirahara*: Supported by JST, PRESTO Grant Number JPMJPR2024, Japan.

Rahul Santhanam: Supported by ESPRC New Horizons Grant No. EP/V048201/1 on “Structure vs Randomness in Algorithms and Computation”, UK.

1 Introduction

In his influential paper on average-case complexity, Impagliazzo [23] discusses five possible “complexity worlds”: Algorithmica, Heuristica, Pessiland, Minicrypt and Cryptomania. Algorithmica is a world where NP is easy in the worst case. Heuristica is a world where NP is hard in the worst case but is easy on average with respect to every samplable distribution. In Pessiland, NP is hard on average but one-way functions don’t exist. This is called the “worst of all possible worlds” in [23], as this is a world in which we can neither solve NP problems efficiently (on average) nor do cryptography. In Minicrypt, one-way functions exist but public-key cryptography does not. Finally, in Cryptomania, public-key cryptosystems exist. Conventional wisdom is that we live in Cryptomania, but given our lack of success in proving unconditional lower bounds against strong circuit classes, even ruling out Algorithmica seems very far beyond our reach, let alone showing that public-key cryptosystems exist.

Despite much effort in the 25 years since Impagliazzo’s work, these five worlds are all possible given our current state of knowledge. We still do not understand the precise relationships between worst-case hardness of NP, average-case hardness of NP and cryptography. Ruling out the existence of Heuristica and Pessiland would be especially significant, as it would allow us to base one-way functions on NP-hardness. There are known limitations to doing this in a black-box way [10, 7, 1, 5].



© Shuichi Hirahara and Rahul Santhanam;

licensed under Creative Commons License CC-BY 4.0

13th Innovations in Theoretical Computer Science Conference (ITCS 2022).

Editor: Mark Braverman; Article No. 85; pp. 85:1–85:25



Leibniz International Proceedings in Informatics

LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Very recently, there have been a series of works [8, 14, 38, 15, 18] using *meta-complexity* to gain a better understanding of the relationship between worst-case hardness and average-case hardness of natural computational problems in situations where black-box reductions are unlikely to exist. Here “meta-complexity” refers to the complexity of computational problems that are themselves about complexity, such as the problem MINKT of deciding whether the t -time-bounded Kolmogorov complexity of a given string x is at most s (where t and s are given in unary) [32], or the problem MCSP of deciding if a Boolean function given by its truth table has circuit complexity at most a given parameter s [30]. Building on work of [8, 19], Hirahara [14, 16] showed an inherently non-black-box reduction from approximating the gap version GapMINKT of MINKT with a logarithmic gap to computing MINKT on average over the uniform distribution in an errorless way. Hirahara’s result is very relevant to the question of whether Heuristica exists – indeed, if GapMINKT were NP-hard, we would be able to rule out Heuristica. We note that NP-hardness of the unrelativized versions of MINKT and MCSP are long-standing open questions, but there has recently been substantial progress [20, 22, 21, 34] on showing hardness of variants of these problems.

In a subsequent paper [15], Hirahara showed that PH is easy on average iff a gap version of MINKT^{PH} is in polynomial time, where MINKT^{PH} denotes the problem of computing the PH-oracle t -time-bounded Kolmogorov complexity of a given string. This result is very relevant to whether a PH-analogue of Heuristica exists, i.e., is it true that if PH is hard in the worst case, then PH is hard on average? By the main result of [15], PH Heuristica would be ruled out by showing that $\text{GapMINKT}^{\text{PH}}$ is NP-hard.

Inspired by the results in [15], we consider the question of whether a PH-analogue of Pessiland exists, and give an unconditional negative answer in both the uniform and non-uniform settings. Our PH analogue of Pessiland is a world where PH is hard on average but PH-computable pseudorandom generators do not exist¹. Note that the PH analogue of Heuristica has not yet been ruled out unconditionally – doing so is essentially equivalent to showing NP-hardness of $\text{GapMINKT}^{\text{PH}}$ [15]². While “PH Pessiland” is not interesting from the cryptographic view point, we believe that our proof techniques constitute progress towards excluding the standard Pessiland, and broaden our understanding of the relationship between hardness and pseudo-randomness. Moreover, as in [15], meta-complexity plays a fundamental role in our results: we show *equivalences* between average-case hardness of PH, existence of PH-computable pseudorandom generators and the average-case hardness of MINKT^{PH} .

We now describe our results more formally.

1.1 Our Results

We unconditionally rule out the existence of “PH Pessiland”. We prove the equivalence between average-case hardness of PH and the existence of PH-computable pseudorandom generator, as well as the DistPH-completeness of MINKT^{PH} .

► **Theorem 17** (Excluding PH Pessiland – the non-uniform case). *For every function $t: \mathbb{N} \rightarrow \mathbb{N}$ such that $n^\epsilon \leq t(n) \leq n^{1/\epsilon}$ for all large $n \in \mathbb{N}$, where $\epsilon > 0$ is an arbitrary constant, the following are equivalent.*

¹ We consider PH-computable pseudorandom generators rather than PH-computable one-way functions, because the notion of a PH-computable one-way function is somewhat ambiguous – it is not a priori clear whether the inverter should be allowed oracle access to the function.

² In [18], a weak PH-hardness of $\text{GapMINKT}^{\text{PH}}$ was proved: If $\text{PH} \not\subseteq \text{DTIME}(2^{O(n/\log n)})$, then $\text{GapMINKT}^{\text{PH}} \notin \text{P}$ and consequently $\text{DistPH} \not\subseteq \text{AvgP}$.

1. $\text{PH} \times \{\mathcal{U}\} \not\subseteq \text{HeurP/poly}$.
2. $\text{DistPH} \not\subseteq \text{HeurP/poly}$.
3. $\text{MINK}^{t,\text{PH}}[n - \log n] \times \{\mathcal{U}\} \not\subseteq \text{HeurP/poly}$.
4. *There exists a PH-computable pseudorandom generator*
 $G = \{G_n : \{0,1\}^n \rightarrow \{0,1\}^{n+1}\}_{n \in \mathbb{N}}$ *secure against P/poly infinitely often.*
5. *For every constant c , there exists a PH-computable pseudorandom generator*

$$G = \left\{ G_n : \{0,1\}^n \rightarrow \{0,1\}^{n^c} \right\}_{n \in \mathbb{N}}$$

secure against P/poly infinitely often.

Here, HeurC denotes the class of distributional problems that admit error-prone heuristic schemes in \mathfrak{C} . DistC denotes the class of distributional problems (L, \mathcal{D}) such that $L \in \mathfrak{C}$ and \mathcal{D} is polynomial-time samplable; see Section 3 for formal definitions.

Previously, it was known that MINKT^{PH} is complete for DistPH in the case of *errorless* average-case complexity. Specifically, the aforementioned previous work [15] presented several equivalent statements on errorless average-case complexity: $\text{DistPH} \subseteq \text{AvgP}$ iff $\text{MINKT}^{\text{PH}} \times \text{PSamp} \subseteq \text{AvgP}$ iff $\text{GapMINKT}^{\text{PH}} \in \text{P}$ iff there exists no PH-computable hitting set generator secure against P and $\text{P} = \text{ZPP}$, where AvgP denotes the class of distributional problems solvable by errorless heuristic schemes. In contrast, our results provide equivalent statements on *error-prone* average-case complexity, denoted by HeurP/poly .

Combined with [15], our results suggest that the relationship between a PH-computable hitting set generator and a PH-computable pseudorandom generator is essentially equivalent to the relationship between errorless average-case complexity and error-prone average-case complexity. The existence of a PH-computable hitting set generator is equivalent to the *errorless* average-case hardness (under the derandomization assumption that $\text{P} = \text{ZPP}$) [15], whereas the existence of a PH-computable pseudorandom generator is equivalent to the *error-prone* average-case hardness.

We also exclude PH Pessiland defined in terms of uniform algorithms. In this case, we show that the average-case hardness of PH is equivalent to the existence of a PH-computable pseudorandom generator that extends its seed by $O(\log n)$ bits. Whether the seed of the PH-computable pseudorandom generator can be extended more remains open. However, we also present an equivalent statement that either there exists a tally hard language $L \subseteq \{1\}^*$ in PH, or there exists a PH-computable pseudorandom generator that extends its seed to n^c bits for an arbitrary constant c .

► **Theorem 27** (Excluding PH Pessiland – the uniform case). *For every function $t : \mathbb{N} \rightarrow \mathbb{N}$ such that $n^\epsilon \leq t(n) \leq n^{1/\epsilon}$ for all large $n \in \mathbb{N}$, where $\epsilon > 0$ is an arbitrary constant, the following are equivalent.*

1. $\text{PH} \times \{\mathcal{U}\} \not\subseteq \text{HeurBPP}$.
2. $\text{MINK}^{t,\text{PH}}[n - \log n] \times \{\mathcal{U}\} \not\subseteq \text{HeurBPP}$.
3. $\text{DistPH} \not\subseteq \text{HeurBPP}$.
4. $\text{PH} \times \text{PSamp}^{\text{PH}} \not\subseteq \text{HeurBPP}$.
5. $\text{PH} \times \{\mathcal{U}\} \not\subseteq \text{Heur}_{n^{-c}}\text{BPP}/\log$ for some constant c .
6. *There exists a PH-computable pseudorandom generator*
 $G = \{G_n : \{0,1\}^n \rightarrow \{0,1\}^{n+1}\}_{n \in \mathbb{N}}$ *secure against BPP infinitely often.*
7. *For every constant c , there exists a PH-computable pseudorandom generator*

$$G = \left\{ G_n : \{0,1\}^n \rightarrow \{0,1\}^{n+c \log n} \right\}_{n \in \mathbb{N}}$$

secure against BPP infinitely often.

8. Either for every constant $c > 1$, there exists a PH-computable pseudorandom generator

$$G = \left\{ G_n : \{0, 1\}^n \rightarrow \{0, 1\}^{n^c} \right\}_{n \in \mathbb{N}}$$

secure against BPP infinitely often, or there exists a tally language L in PH such that $L \notin \text{BPP}$.

2 Overview of Our Proofs

In this section, we present ideas of our proofs.

2.1 Excluding PH Pessiland – The Non-Uniform Case

Here, we explain the proof ideas of showing the following equivalence:

$$\text{PH} \times \{\mathcal{U}\} \not\subseteq \text{HeurP/poly} \iff \exists \text{ PH-computable PRG secure against P/poly.} \quad (1)$$

We first observe that the forward direction of Equation (1) can be proved using the standard construction of complexity-theoretic pseudorandom generators. Specifically, the hardness versus randomness framework developed in, e.g., [47, 4, 36, 2, 24, 28] enables constructing a pseudorandom generator whose security is based on the average-case hardness of a function f : Given a mildly average-case function f , we first convert it into a strongly average-case hard function f' using Yao's XOR lemma [11]. We then use the Nisan–Wigderson pseudorandom generator construction $\text{NW}^{f'}$ based on the strongly average-case hard function f' [36].

This “easy direction” should be contrasted with the case of the standard version of Pessiland. The difficulty of excluding Pessiland is that it is unclear how to construct a polynomial-time-computable pseudorandom generator (or, equivalently, a one-way function [12]) based on the average-case hardness of NP. In contrast, the converse is already known to be true, as was shown by the work of Håstad, Impagliazzo, Levin, and Luby [12] and Impagliazzo and Levin [26].

To prove the backward direction of Equation (1), we employ the ideas developed in [12, 26, 33]. We actually prove that every PH-computable pseudorandom generator is not secure against P/poly under the assumption that MINKT^{PH} is easy on average, using the insights of Liu and Pass [33]. This enables us to establish the “DistPH-completeness” of MINKT^{PH} as well as the backward implication of Equation (1). In [33], the task of inverting one-way functions is reduced to MINKT. In our settings, however, inverting PH-computable one-way functions is not useful: for a PH-computable function G , inverting $G(x)$ (i.e., computing $x' \in G^{-1}(G(x))$ on input $G(x)$) does not necessarily imply that G is not a secure pseudorandom generator because G may not be computed efficiently.

To address this issue, we directly reduce the task of breaking the security of G to MINKT. Let $G : \{0, 1\}^n \rightarrow \{0, 1\}^m$ be a candidate PH-computable pseudorandom generator, where $n < m$. For simplicity, we assume that G is a 2^r -to-1 function for some $r \in \mathbb{N}$. We first make G “entropy-preserving” [33] in the following way:

$$G'(h_1, h_2, x) := (h_1, h_2, h_1(x), h_2(G(x))),$$

where h_1 and h_2 are pairwise independent hash functions such that $|h_1(x)| = r - O(\log n)$ and $|h_2(G(x))| = n - r + O(\log n)$. Let m' denote the output length of G' , i.e., $|h_1| + |h_2| + n + O(\log n)$. Using G' , we can construct an algorithm that distinguishes the output distribution $G(\mathcal{U})$ of G from the uniform distribution in the following two steps:

1. If we truncate the last $O(\log n)$ bits of $G'(h_1, h_2, x)$, then it is statistically close to the uniform distribution. In particular, the entropy of $G'(h_1, h_2, x)$ is large; it is at least $|h_1| + |h_2| + |h_1(x)| + |h_2(G(x))| - O(\log n) \geq m' - O(\log n)$. Liu and Pass [33] showed that this high entropy condition implies that the output distribution of G' (which we denote by $G'(\mathcal{U})$) is approximately dominated by the uniform distribution; thus, the error probability of the heuristic algorithm for MINKT^{PH} with respect to $G'(\mathcal{U})$ is approximately as small as the error probability with respect to the uniform distribution. This enables us to construct a distinguisher for $G'(\mathcal{U})$.
2. The security of G is preserved in the sense that any algorithm that distinguishes $G'(\mathcal{U})$ from \mathcal{U} can be converted into an algorithm that distinguishes $G(\mathcal{U})$ from \mathcal{U} .

Details can be found in Section 4.

2.2 Excluding PH Pessiland – The Uniform Case

Next, we explain the proof ideas of showing a uniform analogue of Equation (1):

$$\text{PH} \times \{\mathcal{U}\} \not\subseteq \text{HeurBPP} \iff \exists \text{ PH-computable PRG secure against BPP}. \quad (2)$$

Unlike the non-uniform case (Equation (1)), it turned out that the backward implication of Equation (2) can be proved by combining previous results in the literature, whereas proving the forward implication is technically more challenging.

We first explain how to prove the backward implication of Equation (2). Let PSamp^{PH} denote the class of distributions samplable with some PH oracle. It is well known (as in, e.g., [46]) that the existence of a PH -computable pseudorandom generator implies the average-case hardness of a distributional problem ($\text{image}(G), \mathcal{D} \in \text{PH} \times \text{PSamp}^{\text{PH}}$, where $\text{image}(G)$ denotes the image of G and $\mathcal{D} := \frac{1}{2}G(\mathcal{U}) + \frac{1}{2}\mathcal{U}$, i.e., the distribution which is equal to $G(\mathcal{U})$ with probability 1/2 and \mathcal{U} with probability 1/2. In the case of uniform algorithms, it is possible to show

$$\text{PH} \times \{\mathcal{U}\} \subseteq \text{HeurBPP} \iff \text{PH} \times \text{PSamp}^{\text{PH}} \subseteq \text{HeurBPP},$$

from which the backward implication of Equation (2) follows. The proof idea of this equivalence is as follows: We first use the theorem of Impagliazzo and Levin [26] to show that $\text{PH} \times \{\mathcal{U}\} \subseteq \text{HeurBPP}$ if and only if $\text{DistPH} := \text{PH} \times \text{PSamp} \subseteq \text{HeurBPP}$. We then use the ideas developed in [3, 39] to show that $\text{DistPH} \subseteq \text{HeurBPP}$ implies $\text{PH} \times \text{PSamp}^{\text{PH}} \subseteq \text{HeurBPP}$: Let $(L, \mathcal{D}) \in \text{PH} \times \text{PSamp}^{\text{PH}}$ be an arbitrary distributional problem. Then there exists a PH -computable sampler S that, given a coin flip sequence r , samples a string distributed according to \mathcal{D} . We then regard the problem of computing the output of S on input a random coin flip sequence r as a distributional problem in DistPH , which can be solved by an efficient heuristic algorithm A on average (using the assumption that $\text{DistPH} \subseteq \text{HeurBPP}$). Thus, the PH -computable distribution \mathcal{D} can be approximated by an efficient algorithm A and we can regard A as a polynomial-time samplable distribution. We then consider the distributional problem $(L, A) \in \text{DistPH} \subseteq \text{HeurBPP}$, whose efficient heuristic scheme can be translated into a heuristic scheme for $(L, \mathcal{D}) \in \text{PH} \times \text{PSamp}^{\text{PH}}$ using the fact that \mathcal{D} is approximated by A . We emphasize that the same proof idea does not work in the non-uniform setting because we exploit the uniformity of A to show that $A \in \text{PSamp}$.

Now, we explain how to prove the forward implication of Equation (1). As in the non-uniform setting, we again use the hardness versus randomness framework that enables converting the average-case hardness of PH to the existence of a PH -computable pseudorandom generator. However, there is an inherent difficulty of applying the framework in the uniform setting: Trevisan and Vadhan [41] showed that any “security reduction” for a black-box construction of pseudorandom generators $G^f : \{0, 1\}^n \rightarrow \{0, 1\}^{n+k}$ based on a hard problem

f requires approximately k bits of non-uniform advice; thus, we need non-uniform hardness for f to construct a secure pseudorandom generator G^f . This issue can be circumvented for a polynomial-time computable pseudorandom generator G , in which case the non-uniform advice can be computed by evaluating G . Similarly, the issue can be circumvented when constructing PSPACE-computable pseudorandom generators from the average-case hardness of PSPACE, by using the equivalence between the worst-case and average-case hardness of PSPACE together with the downward self-reducibility of PSPACE [29, 41]; however, PH does not admit a similar worst-case-to-average-case connection [45, 44].

To address the issue of non-uniformity in the security reduction, we show that $O(\log n)$ bits of non-uniformity can be removed for PH. Specifically, following [41], we first introduce a class $\text{HeurBPP}/\log$ of randomized heuristic algorithms that take $O(\log n)$ bits of advice that can depend on the random bits used by the randomized algorithm (but not on the input). This notion of powerful advice enables capturing the non-uniformity introduced by a security reduction.³ Using the hardness versus randomness framework together with an almost uniform version of Yao’s XOR lemma (established by Impagliazzo, Jaiswal, Kabanets, and Wigderson [25]), we construct a PH-computable pseudorandom generator $G: \{0,1\}^n \rightarrow \{0,1\}^{n+O(\log n)}$ based on the assumption that $\text{DistPH} \not\subseteq \text{HeurBPP}/\log$. Then we prove that the $O(\log n)$ bits of advice can be removed:

► **Theorem 22.** *If $\text{DistPH} \subseteq \text{Heur}_{n-c}\text{BPP}/\log$ for every constant $c > 0$, then $\text{DistPH} \subseteq \text{HeurBPP}$.*

Interestingly, the proof of Theorem 22 is non-black-box in the sense that it is unclear if the result can be generalized to the implication $\text{DistPH} \subseteq \text{HeurBPP}^R/\log \implies \text{DistPH} \subseteq \text{HeurBPP}^R$ for every oracle R . There is a general framework called a selector, whose existence for a paddable language L exactly characterizes the condition that $L \in \text{BPP}^R/\log \iff L \in \text{BPP}^R$ for every oracle R [13]. A *selector* for L is a randomized polynomial-time oracle algorithm that computes L given oracle access to two oracles one of which is guaranteed to be equal to L . It is known that any Σ_k^P -complete problem admits a selector for every $k \in \mathbb{N}$ [13]; thus, we have $\text{PH} \subseteq \text{BPP}^R/\log \iff \text{PH} \subseteq \text{BPP}^R$ for every oracle R . However, the selector makes adaptive queries to an oracle, which makes it difficult to apply a similar black-box proof technique to the case of heuristic algorithms. Consequently, we failed to capture Theorem 22 using the framework of selectors because of the non-black-box use of the hypothetical algorithm for PH.

We now sketch the ideas of Theorem 22. It is useful to regard an algorithm with $O(\log n)$ bits of advice as polynomially many algorithms $A_1, \dots, A_{n^{O(1)}}$ one of which is guaranteed to be close to DistPH , where each algorithm is assigned one advice string. Our goal is to compute DistPH using such algorithms. We prove $\text{Dist}\Sigma_k^P \subseteq \text{HeurBPP}/\log \implies \text{Dist}\Sigma_k^P \subseteq \text{HeurBPP}$ by induction on $k \geq 1$. Using the search-to-decision reduction of Valiant and Vazirani [43] and Ben-David, Chor, Goldreich and Luby [3], we find a candidate certificate for Σ_k^P using each algorithm A_i . To verify the correctness of a certificate (which is a problem in Π_{k-1}^P), we use the induction hypothesis to obtain a HeurBPP algorithm. (The distribution on which we verify the correctness of a certificate is induced by the algorithm A_i , which makes the proof non-black-box.) By choosing the success probability of the HeurBPP algorithm sufficiently small compared to the number $n^{O(1)}$ of advice strings, we can verify the correctness of all the $n^{O(1)}$ certificates with high probability.

³ Note that the standard Karp–Lipton advice [31] can depend on the length of an input but cannot depend on random bits used by randomized algorithms.

Finally, we consider the question of whether the seed of a PH-computable pseudorandom generator against uniform algorithms can be extended more. As mentioned before, the impossibility result of [41] suggests that a large amount of non-uniformity is required to construct a pseudorandom generator with large stretch. We will show in Lemma 26 that the non-uniform and uniform average-case complexities of PH are in fact equivalent if every tally language in PH is in BPP. The proof idea is inspired by the work of Pavan, Santhanam, and Vinodchandran [37]: the problem of finding the lexicographically first circuit that computes PH on average can be formulated as a tally language in PH, which enables computing the circuit by a uniform algorithm.

Details can be found in Section 5.

3 Preliminaries

Notation

\mathcal{U}_n denotes the uniform distribution over $\{0, 1\}^n$. Let $\mathcal{U} = \{\mathcal{U}_n\}_{n \in \mathbb{N}}$. We often identify a language $L \subseteq \{0, 1\}^*$ with its characteristic function $L: \{0, 1\}^* \rightarrow \{0, 1\}$. $[n]$ denotes $\{1, 2, \dots, n\}$ for $n \in \mathbb{N}$. Let $\langle \cdot, \cdot \rangle: \mathbb{N}^2 \rightarrow \mathbb{N}$ be a bijection that is defined as, e.g., $\langle a, b \rangle = \sum_{i=0}^{a+b} i + a$. Similarly, let $\langle a, b, c \rangle := \langle \langle a, b \rangle, c \rangle$.

For a distribution \mathcal{D} , $H(\mathcal{D})$ denotes the Shannon entropy of \mathcal{D} ; $H_\infty(\mathcal{D})$ denotes the min-entropy of \mathcal{D} . For $x \in \text{supp}(\mathcal{D})$, let $\mathcal{D}(x)$ denote $\Pr_{X \sim \mathcal{D}}[X = x]$. For two distributions \mathcal{D} and \mathcal{D}' , the statistical distance between \mathcal{D} and \mathcal{D}' is denoted by $\|\mathcal{D} - \mathcal{D}'\|$.

3.1 Pseudorandomness

We provide the definition of a PH-computable pseudorandom generator below.

► **Definition 1.** Let \mathfrak{C} be a class of algorithms. A function $G = \{G_n : \{0, 1\}^{s(n)} \rightarrow \{0, 1\}^{t(n)}\}_{n \in \mathbb{N}}$ is said to be a pseudorandom generator (PRG) secure against i.o. \mathfrak{C} (or secure against \mathfrak{C} infinitely often) if

1. the unary representations of $s(n)$ and $t(n)$ are computable in polynomial time given 1^n as input,
2. $s(n) < t(n)$ for all large $n \in \mathbb{N}$, and
3. for every $A \in \mathfrak{C}$ and any polynomial p , there exists an infinite set $I \subseteq \mathbb{N}$ such that

$$\left| \Pr_{w \sim \{0, 1\}^{t(n)}, A} [A(w) = 1] - \Pr_{z \sim \{0, 1\}^{s(n)}, A} [A(G(z)) = 1] \right| \leq \frac{1}{p(n)}$$

for any $n \in I$, where the probabilities are taken over w and z as well as the internal randomness of A (if A is a randomized algorithm).

The function s is referred to as the seed length of G . For a complexity class \mathfrak{D} , we say that G is \mathfrak{D} -computable if, given a string $z \in \{0, 1\}^{s(n)}$ and integers $n \in \mathbb{N}$, $t(n)$ and $i \in [n]$ represented in unary, the i -th bit of $G_n(z)$ can be computed in \mathfrak{D} .

3.2 Average-Case Complexity

Here we review some standard notions of average-case complexity. We refer the readers to an excellent survey of Bogdanov and Trevisan [6] for a detailed exposition on the theory of average-case complexity.

► **Definition 2** (Polynomial-Time Samplable). *For an oracle A , we say that a family $\mathcal{D} = \{\mathcal{D}_n\}_{n \in \mathbb{N}}$ of distributions is polynomial-time samplable with oracle A if there exist an oracle polynomial-time algorithm M and a polynomial p such that, for every $n \in \mathbb{N}$ and every $x \in \{0, 1\}^*$,*

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

For a function $t: \mathbb{N} \rightarrow \mathbb{N}$, if M runs in time $t(n)$ on input $(1^n, -)$ for every $n \in \mathbb{N}$, we say that \mathcal{D} is $t(n)$ -time samplable. Let PSamp^A denote the class of polynomial-time samplable families of distributions with oracle A . We omit the superscript A if $A = \emptyset$. For a complexity class \mathfrak{C} , let $\text{PSamp}^\mathfrak{C}$ denote $\bigcup_{A \in \mathfrak{C}} \text{PSamp}^A$. Let $\text{Dist}\mathfrak{C}$ denote $\mathfrak{C} \times \text{PSamp}$.

► **Definition 3** (Error-prone Heuristic Scheme [23, 6]). *An algorithm A is said to be an error-prone heuristic scheme for a distributional problem (L, \mathcal{D}) if for every $n \in \mathbb{N}$ and $\delta^{-1} \in \mathbb{N}$,*

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

where the probability is taken over $x \sim \mathcal{D}_n$ as well as a coin flip sequence of A (if A is a randomized algorithm). The class of distributional problems that admit randomized polynomial-time error-prone heuristic schemes is denoted by HeurBPP . For an oracle R , HeurBPP^R denotes the class of distributional problems for which there exist R -oracle randomized polynomial-time error-prone heuristic schemes. Similarly, HeurP/poly denotes the class of distributional problems for which there exist error-prone heuristic schemes of polynomial-size circuits. For notational simplicity, we often write $A(x; n, \delta)$ instead of $A(x; 1^n, 1^{\delta^{-1}})$. For a function $\delta: \mathbb{N} \rightarrow [0, 1]$ and a class $\mathfrak{C} \in \{\text{BPP}, \text{P/poly}\}$, let $\text{Heur}_\delta \mathfrak{C}$ denote the class of distributional problems for which there exists an algorithm $A \in \mathfrak{C}$ such that

$$\Pr_{x \sim \mathcal{D}_n, A} [A(x; n) \neq L(x)] \leq \delta(n).$$

► **Lemma 4.** *For every oracle A and every class $\mathfrak{C} \in \{\text{BPP}, \text{P/poly}\}$, if $\text{NP}^A \times \{\mathcal{U}\} \subseteq \text{Heur}_\epsilon \mathfrak{C}$ for $\epsilon(n) = 1/n$, then $\text{Dist}\text{NP}^A \subseteq \text{Heur} \mathfrak{C}$.*

Proof Sketch. Using a simple padding argument as in [23, 6], it can be shown that $\text{NP}^A \times \{\mathcal{U}\} \subseteq \text{Heur}_\epsilon \mathfrak{C}$ if and only if $\text{NP}^A \times \{\mathcal{U}\} \subseteq \text{Heur} \mathfrak{C}$. Impagliazzo and Levin [26] (see also [6]) showed that $\text{NP}^A \times \{\mathcal{U}\} \subseteq \text{Heur} \mathfrak{C}$ implies $\text{Dist}\text{NP}^A \subseteq \text{Heur} \mathfrak{C}$. ◀

3.3 Kolmogorov Complexity

The t -time-bounded Kolmogorov complexity $K^t(x)$ of x is defined as follows.

► **Definition 5.** *For a string $x \in \{0, 1\}^*$, an oracle $A \subseteq \{0, 1\}^*$, and a time bound $t \in \mathbb{N} \cup \{\infty\}$,*

$$K^{t,A}(x) := \min \{ |d| \mid U^{d,A}(i) \text{ outputs } x_i \text{ in time } t \text{ for each } i \in [|x| + 1] \}.$$

Here, $U^{d,A}(i)$ means the output of the universal Turing machine given random access to d and A and i as input; x_i denotes the i th bit of x if $i \leq |x|$ and \perp otherwise. We omit the superscript A if $A = \emptyset$ and the superscript t if $t = \infty$.

We consider the problem of computing time-bounded Kolmogorov complexity, which is called MINKT [32].

► **Definition 6.** For an oracle A and functions $s: \mathbb{N} \rightarrow \mathbb{N}$ and $t: \mathbb{N} \rightarrow \mathbb{N}$,

$$\text{MINK}^{t,A}[s] := \{x \in \{0,1\}^* \mid K^{t,A}(x) \leq s(|x|)\}.$$

We omit the superscript A if $A = \emptyset$. For a complexity class \mathfrak{C} , we define

$$\text{MINK}^{t,\mathfrak{C}}[s] := \left\{ \text{MINK}^{t,A}[s] \mid A \in \mathfrak{C} \right\}.$$

The following simple fact shows that the number of YES instances of MINKT is small.

► **Fact 7.** For integers $n \in \mathbb{N}$ and $\theta \in \mathbb{N}$ and every oracle A ,

$$\Pr_{x \sim \{0,1\}^n} [K^A(x) \leq \theta] \leq 2^{\theta+1-n}.$$

Proof Sketch. The number of descriptions d of length at most θ is at most $2^{\theta+1}$. ◀

3.4 Nonadaptive Oracle Machine

A randomized polynomial-time *nonadaptive* oracle machine $M^{(\cdot)}$ is a polynomial-time oracle Turing machine M such that there exists a polynomial-time-computable function Q such that, for any input $x \in \{0,1\}^*$, any coin flip sequence $r \in \{0,1\}^*$ and every oracle O , any query made by M^O on input $(x; r)$ is in the list $Q(x; r)$ of strings. For a fixed input $x \in \{0,1\}^*$ and $i \in \mathbb{N}$, let $Q(x; r)_i$ denote the i -th string in $Q(x; r)$. We may assume without loss of generality that the marginal distribution of $Q(x; r)_i$ is identical to that of $Q(x; r)_j$ for every $(i, j) \in \mathbb{N}^2$ over a random choice of r ; indeed, one can randomly permute the list $Q(x; r)$. The distribution of $Q(x; r)_1$ over a random choice of r is said to be the *query distribution* of M on input x . For a query distribution $Q(x)$ and a distribution \mathcal{D} , let $Q \circ \mathcal{D}$ denote the distribution from which a random sample q is generated by sampling $x \sim \mathcal{D}$ and $q \sim Q(x)$. For a family $\mathcal{D} = \{\mathcal{D}_n\}_{n \in \mathbb{N}}$, let $Q \circ \mathcal{D}$ denote $\{Q \circ \mathcal{D}_n\}_{n \in \mathbb{N}}$.

4 Excluding Non-Uniform PH Pessiland

In this section, we exclude the non-uniform version of PH Pessiland.

4.1 Non-Uniform Hardness from Pseudorandom Generators

We prove that if there exists a \mathfrak{C} -computable pseudorandom generator, then $\text{NP}^\mathfrak{C}$ is average-case hard with respect to the uniform distribution.

► **Theorem 8.** Let \mathfrak{C} be any complexity class and $t: \mathbb{N} \rightarrow \mathbb{N}$ be a function such that $t(n) = n^{\Omega(1)}$. If $\text{MINK}^{t,\mathfrak{C}}[n - \log n] \times \{\mathcal{U}\} \subseteq \text{HeurP/poly}$, then there exists no \mathfrak{C} -computable pseudorandom generator secure against P/poly infinitely often.

Since $\text{MINK}^{t,\text{PH}}[s] \subseteq \text{NP}^{\text{PH}} = \text{PH}$ for every $t(n) = n^{O(1)}$ and every s , Theorem 8 implies the following corollary.

► **Corollary 9.** For every $t: \mathbb{N} \rightarrow \mathbb{N}$ such that $n^{\Omega(1)} \leq t(n) \leq n^{O(1)}$, if $\text{MINK}^{t,\mathfrak{C}}[n - \log n] \times \{\mathcal{U}\} \subseteq \text{HeurP/poly}$, then there exists no PH -computable pseudorandom generator secure against P/poly infinitely often.

The remainder of this subsection is devoted to proving Theorem 8. We start with a simple observation.

► **Lemma 10.** *For every complexity class \mathfrak{C} closed under polynomial-time many-one reductions and every constant $\epsilon \in (0, 1]$, we have Item 1 \implies Item 2 \iff Item 3 in the following list:*

1. *There exists a \mathfrak{C} -computable pseudorandom generator*

$$G = \left\{ G_n : \{0, 1\}^{s(n)} \rightarrow \{0, 1\}^{t(n)} \right\}_{n \in \mathbb{N}}$$

secure against P/poly infinitely often.

2. *There exists a \mathfrak{C} -computable pseudorandom generator*

$$G = \left\{ G_n : \{0, 1\}^n \rightarrow \{0, 1\}^{n+1} \right\}_{n \in \mathbb{N}}$$

secure against P/poly infinitely often.

3. *For every constant $\gamma > 0$, there exists a \mathfrak{C} -computable pseudorandom generator*

$$G = \left\{ G_n : \{0, 1\}^n \rightarrow \{0, 1\}^{n+\gamma \log n} \right\}_{n \in \mathbb{N}}$$

secure against P/poly infinitely often.

Proof. (Item 1 \Rightarrow 2) Define $G'_n : \{0, 1\}^{s(n)} \rightarrow \{0, 1\}^{s(n)+1}$ so that $G'_n(z)$ is the first $s(n) + 1$ bits of $G_n(z)$ for every $z \in \{0, 1\}^{s(n)}$. It is easy to observe that the security of G implies the security of $G' := \left\{ G'_n : \{0, 1\}^{s(n)} \rightarrow \{0, 1\}^{s(n)+1} \right\}_{n \in \mathbb{N}}$. Similarly, we obtain the implication 3 \Rightarrow 2.

(Item 2 \Rightarrow 3) Let $k = k(n)$ be a parameter chosen later. For each $n \in \mathbb{N}$, define a function

$$G_n^k : \{0, 1\}^{kn} \rightarrow \{0, 1\}^{kn+k}$$

so that

$$G_n^k(z_1, \dots, z_k) := G_n(z_1) \cdots G_n(z_k)$$

for every $z_1, \dots, z_k \in \{0, 1\}^n$. By a standard hybrid argument (see, e.g., [42]), a distinguisher for G_n^k can be converted into a distinguisher for G_n for every $n \in \mathbb{N}$; therefore, $G^k := \left\{ G_n^k \right\}_{n \in \mathbb{N}}$ is a pseudorandom generator secure against P/poly . It is easy to observe that G^k is computable in \mathfrak{C} . We choose $k = O(\log n)$ so that $nk + \gamma \log(nk) \leq nk + k$; then, we obtain a \mathfrak{C} -computable pseudorandom generator

$$G' = \left\{ G'_n : \{0, 1\}^{nk} \rightarrow \{0, 1\}^{nk + \gamma \log(nk)} \right\}_{n \in \mathbb{N}}$$

by truncating the output of G_n^k to $nk + \gamma \log(nk)$ bits. ◀

In light of Lemma 10, in order to prove Theorem 8, it suffices to prove that for some constant $\gamma > 0$, for any \mathfrak{C} -computable family G of functions such that

$$G = \left\{ G_n : \{0, 1\}^n \rightarrow \{0, 1\}^{n+\gamma \log n} \right\}_{n \in \mathbb{N}},$$

there exists a P/poly algorithm that distinguishes $G(\mathcal{U}_n)$ from $\mathcal{U}_{n+\gamma \log n}$ for all large inputs. To this end, we show that G can be converted into a pseudorandom generator that does not lose the entropy of its seed while retaining the security of G . We use a standard pairwise independent hash family, whose property is described below.

► **Lemma 11 (Leftover hash lemma [12]).** *There exists a polynomial-time-computable family*

$$\{\text{Ext}_{n,m} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m\}_{n,m \in \mathbb{N}}$$

such that $d \leq \text{poly}(n, m)$ and for every $\epsilon > 0$, every $m' \leq m$ and every distribution X on $\{0, 1\}^n$, if $m' \leq H_\infty(X) - 2\log(1/\epsilon)$, then

$$\|(\mathcal{U}_d, \text{Ext}_{n,m}(X, \mathcal{U}_d)|_{m'}) - (\mathcal{U}_d, \mathcal{U}_{m'})\| \leq \epsilon,$$

where $z|_{m'}$ denotes the first $\max\{0, m'\}$ bits of z .

Proof Sketch. The function $\text{Ext}_{n,m}: \{0, 1\}^n \times \{0, 1\}^{(n+1)m} \rightarrow \{0, 1\}^m$ is defined as follows: For every $x \in \{0, 1\}^n \cong \text{GF}(2)^n$,

$$\text{Ext}(x, (A, b)) := A \cdot x + b,$$

where A is an $m \times n$ matrix A over $\text{GF}(2)$ and $b \in \text{GF}(2)^m$. It is known that $\text{Ext}_{n,m}$ is a pairwise independent hash family [9] and satisfies the property of an extractor [12]. \blacktriangleleft

The following lemma shows that the entropy of a random variable close to the uniform distribution is large.

► **Lemma 12** ([35]). *For any random variable Y over $\{0, 1\}^m$ and any $\epsilon \in [0, 1]$, if $\|Y - \mathcal{U}_m\| \leq \epsilon$, then $H(Y) \geq m(1 - \epsilon) - 1$.*

We are now ready to prove Theorem 15.

Proof of Theorem 15. Let $G = \{G_n: \{0, 1\}^n \rightarrow \{0, 1\}^{n+\gamma \log n}\}_{n \in \mathbb{N}}$ be a family of functions, where γ is a large constant chosen later. By Lemma 10, it suffices to show that for some constant γ , there exists a P/poly algorithm that distinguishes $G(\mathcal{U}_n)$ from $\mathcal{U}_{n+\gamma \log n}$.

Let $c < \gamma$ be a constant chosen later. For every $n \in \mathbb{N}$ and every $i \leq n$, we define $G'_{n,i}$ as follows:

$$G'_{n,i}(z_1, z_2, x) := (z_1, z_2, \text{Ext}(x, z_1)|_i, \text{Ext}(G(x), z_2)|_{n-i+(\gamma-c)\log n}),$$

where Ext is the function from Lemma 11, $x \in \{0, 1\}^n$, and “,” denotes the concatenation of strings. Let $s = s(n) := |z_1| + |z_2|$. Note that the function $G'_{n,i}$ takes a seed of length $s+n$ and outputs a string of length $s+n+\gamma' \log n$, where $\gamma' := \gamma - c$.

Fix an integer $n \in \mathbb{N}$. Consider a set

$$S_r := \{x \in \{0, 1\}^n \mid 2^{r-1} \leq |G_n^{-1}(G_n(x))| \leq 2^r\}.$$

Since $\{0, 1\}^n = \bigcup_{r \in [n]} S_r$, there exists an index $r \in [n]$ such that $\Pr_{x \sim \{0, 1\}^n}[x \in S_r] \geq 1/n$. Let $r = r(n)$ denote such an index. Let $i = i(n) := \max\{0, r(n) - c \log n\}$. Following [33], we prove that G' is “entropy-preserving” in the following sense.

► **Claim 13.** For every $n \in \mathbb{N}$, let Y be the random variable $G'_{n,i(n)}(z_1, z_2, x)$ where (z_1, z_2) is sampled from \mathcal{U}_s and x is sampled uniformly from $S_{r(n)}$. Then, Y satisfies the following properties:

1. $H(Y) \geq s + n - (2c + 1) \log n - 2$.
2. There exists a randomized polynomial-size oracle circuit $A^{(+)}$ such that for every $\epsilon > 0$ and for every oracle D such that

$$\Pr[D(Y) = 1] - \Pr[D(\mathcal{U}_{s+n+\gamma' \log n}) = 1] \geq \epsilon,$$

it holds that

$$\Pr_A[A^D(G(\mathcal{U}_n)) = 1] - \Pr_A[A^D(\mathcal{U}_{n+\gamma \log n}) = 1] \geq \epsilon - 3n^{-c/2}.$$

85:12 Excluding PH Pessiland

Proof. Let X denote the random variable distributed uniformly on $S_{r(n)}$ (i.e., $X \sim S_{r(n)}$). To prove Item 1, let Y' be the random variable such that

$$Y' \equiv (z_1, z_2, \text{Ext}(X, z_1)|_i, \text{Ext}(G_n(X), z_2)|_{n-i-(2c+1)\log n}),$$

where $(z_1, z_2) \sim \{0, 1\}^s$ and $i := i(n)$. Since Y' is a prefix of Y , we have $H(Y') \leq H(Y)$. Below, we prove that Y' is statistically close to the uniform distribution, which implies that the entropy of Y' is large.

Let Z_1 be the random variable such that

$$Z_1 \equiv (z_1, z_2, \mathcal{U}_i, \text{Ext}(G_n(X), z_2)|_{n-i-(2c+1)\log n}),$$

where $(z_1, z_2) \sim \{0, 1\}^s$. If we fix $G_n(X)$ to an arbitrary value $g \in \text{supp}(G_n(X))$, the entropy of $X \sim S_r$ is at least 2^{r-1} ; that is, $H_\infty(X | G_n(X) = g) \geq r - 1$. By Lemma 11, the statistical distance between Y' and Z_1 conditioned on $G_n(X) = g$ is at most $2n^{-c/2}$. By taking an average over all $g \in \text{supp}(G_n(X))$, we obtain

$$\|Y' - Z_1\| \leq 2n^{-c/2}. \quad (3)$$

Let Z_2 be the random variable such that

$$Z_2 := (z_1, z_2, \mathcal{U}_i, \mathcal{U}_{n'}),$$

where $(z_1, z_2) \sim \{0, 1\}^s$ and $n' := \max \{0, n - i - (2c + 1)\log n\}$. Since $\Pr_X [G_n(X) = g] = \frac{|G_n^{-1}(g)|}{|S_r|} \leq 2^r \cdot 2^{-(n-\log n)}$, we have

$$H_\infty(G_n(X)) = \min_{g \in \text{supp}(G_n(X))} \left(-\log \Pr_X [G_n(X) = g] \right) \geq n - \log n - r.$$

We thus have $n - i - (2c + 1)\log n \leq n - \log n - r - c\log n \leq H_\infty(G_n(X)) - c\log n$. Therefore, by Lemma 11, we obtain

$$\|Z_1 - Z_2\| \leq n^{-c/2}. \quad (4)$$

By combining Equations (3) and (4), we have $\|Y' - Z_2\| \leq 3n^{-c/2}$. Since Z_2 is identical to the uniform distribution over $\{0, 1\}^{s+i+n'}$, applying Lemma 12 to Y' and Z_2 , we obtain

$$H(Y') \geq (s + i + n')(1 - 3n^{-c/2}) - 1 \geq s + n - (2c + 1)\log n - 2,$$

where the last inequality holds by choosing c large enough. This completes the proof of Item 1.

To prove Item 2, let $D: \{0, 1\}^{s+n+\gamma' \log n} \rightarrow \{0, 1\}$ be an oracle such that

$$\mathbb{E}[D(Y) - D(\mathcal{U}_{s+n+\gamma' \log n})] \geq \epsilon.$$

Let Z'_1 be the random variable such that

$$Z'_1 := (z_1, z_2, \mathcal{U}_i, \text{Ext}(G_n(x), z_2)|_{n-i+\gamma' \log n}).$$

Using the same argument for showing $\|Y - Z_1\| \leq 2n^{-c/2}$, we have $\|Y - Z'_1\| \leq 2n^{-c/2}$; therefore, we obtain

$$\mathbb{E}[D(Z'_1) - D(\mathcal{U}_{s+n+\gamma' \log n})] \geq \epsilon - 2n^{-c/2}.$$

Consider an oracle algorithm A^D that takes $w \in \{0, 1\}^{n+\gamma \log n}$ as input, picks $(z_1, z_2) \sim \{0, 1\}^s$, and outputs

$$D(z_1, z_2, \mathcal{U}_i, \text{Ext}(w, z_2)|_{n-i+(\gamma-c)\log n}).$$

Observe that $\mathbb{E}_{A,X} [A^D(G_n(X))] = \mathbb{E}[D(Z'_1)]$, where the first expectation is taken over the internal randomness of A as well as $X \sim S_r$. We also have

$$\left| \mathbb{E}_A [A^D(\mathcal{U}_{n+\gamma \log n})] - \mathbb{E}[D(\mathcal{U}_{s+n+\gamma \log n})] \right| \leq n^{-c/2}.$$

because $\text{Ext}(\mathcal{U}_{n+\gamma \log n}, z_2)|_{n-i+(\gamma-c)\log n}$ is statistically close to the uniform distribution by Lemma 11. Therefore, we obtain

$$\mathbb{E}_A [A^D(G(\mathcal{U}_n)) - A^D(\mathcal{U}_{n+\gamma \log n})] \geq \epsilon - 3n^{-c/2},$$

where the expectation is taken over $\mathcal{U}_n, \mathcal{U}_{n+\gamma \log n}$, and the internal randomness of A . \triangleleft

\triangleright **Claim 14.** Let $t(n) = n^{\Omega(1)}$ be a function. Assume that $\text{MINK}^{t,\mathfrak{C}}[n - \log n] \times \{\mathcal{U}\} \subseteq \text{Heur}_\epsilon \mathsf{P/poly}$ for every $\epsilon: \mathbb{N} \rightarrow [0, 1]$ such that $1/\epsilon(n) = n^{O(1)}$. Then, there exists a family $\{\mathcal{D}_n\}_{n \in \mathbb{N}}$ of polynomial-size circuits such that for all large $n \in \mathbb{N}$,

$$\Pr[D_n(Y) = 1] - \Pr[D_n(\mathcal{U}_m) = 1] \geq \frac{1}{n},$$

where $Y = G'_{n,i(n)}(\mathcal{U}_{s(n)}, X)$ is the random variable from Claim 13 and $X \sim S_{r(n)}$.

Proof. We prove this claim by combining a simple padding argument with [33]. Let $p(n)$ be a polynomial chosen later, which determines the amount of padding.

Since $G = \{G_n\}_{n \in \mathbb{N}}$ is \mathfrak{C} -computable, there exists an oracle $B \in \mathfrak{C}$ such that each bit of $G_n(z)$ can be computed in polynomial time given oracle access to B . Let $L := \text{MINK}^{t,B}[n - \log n]$ and let $D \in \mathsf{P/poly}$ be a heuristic algorithm for $(L, \mathcal{U}) \in \text{Heur}_\epsilon \mathsf{P/poly}$. Fix an integer $n \in \mathbb{N}$ and let $s := s(n)$ and $m = m(n) := s + n + \gamma' \log n + p(n)$. Let $D_n(w) := D(w; m(n))$ for every $w \in \{0, 1\}^{m(n)}$ and every $n \in \mathbb{N}$. Below, we prove that $\{\mathcal{D}_n\}_{n \in \mathbb{N}}$ distinguishes $G'_{n,i}(\mathcal{U}_s, X) \cdot \mathcal{U}_{p(n)}$ from \mathcal{U}_m .

We first consider the distribution \mathcal{U}_m . Observe that

$$\Pr[D_n(\mathcal{U}_m) = 1] \leq \Pr[\text{K}^{t,B}(\mathcal{U}_m) \leq m - \log m] + \Pr_{w \sim \{0,1\}^m} [D_n(w) \neq L(w)] \leq \frac{2}{m} + \epsilon(m), \quad (5)$$

where the last inequality follows from Fact 7.

Next, we prove that $\Pr[D_n(G'_{n,i}(\mathcal{U}_s, X) \cdot \mathcal{U}_{p(n)}) = 1]$ is large, where $X \sim S_{r(n)}$. We claim that $G'_{n,i}(z, x) \cdot w \in L$ for every $z \in \{0, 1\}^s$, every $x \in \{0, 1\}^n$, and every $w \in \{0, 1\}^{p(n)}$. Since $G'_{n,i}(z, x)$ can be described by $z \in \{0, 1\}^s$, $x \in \{0, 1\}^n$ and integers $n \in \mathbb{N}$ and $i \in \mathbb{N}$ in polynomial time given oracle access to B , each bit of the padded string $G'_{n,i}(z, x) \cdot w \in \{0, 1\}^m$ can be described in time $t(m)$ given oracle access to B by choosing the polynomial $p(n)$ large enough; thus, we obtain

$$\text{K}^{t,B}(G'_{n,i}(z, x) \cdot w) \leq s + n + |w| + O(\log n) \leq m - \log m,$$

where the last inequality holds by choosing γ large enough. It follows that $G'_{n,i}(z, x) \cdot w \in L$. This means that every string $y \in \text{supp}(Y \cdot \mathcal{U}_{p(n)})$ is a YES instance of L ; thus, in order to prove that $\Pr[D_n(Y \cdot \mathcal{U}_{p(n)}) = 1]$ is large, it suffices to show that the error probability of D_n is small with respect to the input distribution $Y' := Y \cdot \mathcal{U}_{p(n)} = G'_{n,i}(\mathcal{U}_s, X) \cdot \mathcal{U}_{p(n)}$.

To this end, we use the argument of domination: The idea is that the condition that the entropy of Y' is large implies that the distribution Y' is approximately dominated by the uniform distribution. To be more specific, let $I(y) := -\log \Pr[Y' = y]$ for $y \in \text{supp}(Y')$. By Item 1 of Claim 13, we have $\mathbb{E}[I(Y')] = H(Y) + H(\mathcal{U}_{p(n)}) \geq s + n - (2c+1)\log n - 2 + p(n) =: \theta$. By an averaging argument, we obtain $\Pr[I(Y') \geq \theta - 4] \geq \frac{4}{m}$. Therefore, we have

$$\Pr[D_n(Y') \neq L(Y')] \leq 1 - \frac{4}{m} + \Pr[D_n(Y') \neq L(Y') \text{ and } I(Y') \geq \theta - 4].$$

Now,

$$\begin{aligned} & \Pr[D_n(Y') \neq L(Y') \text{ and } I(Y') \geq \theta - 4] \\ & \leq \sum_{y \in \text{supp}(Y')} \mathbb{1}[D_n(y) \neq L(y)] \cdot \Pr[Y' = y] \\ & \leq \sum_{y \in \{0,1\}^m} \mathbb{1}[D_n(y) \neq L(y)] \cdot 2^{-(\theta-4)} \\ & \leq \Pr[D_n(\mathcal{U}_m) \neq L(\mathcal{U}_m)] \cdot 2^{m-\theta+4} \\ & \leq \epsilon(m) \cdot 2^{(2c+1+\gamma') \log n + 6}. \end{aligned}$$

Since $y \in L$ for every $y \in \text{supp}(Y')$, it follows that

$$\Pr[D_n(Y') = 1] = \Pr[D_n(Y') = L(Y')] \geq \frac{4}{m} - \epsilon(m) \cdot 2^6 \cdot n^{2c+1+\gamma'}. \quad (6)$$

By Equations (5) and (6), we obtain

$$\Pr[D_n(Y') = 1] - \Pr[D_n(\mathcal{U}_m) = 1] \geq \frac{2}{m} - \epsilon(m) \cdot 2^6 \cdot n^{2c+1+\gamma'} - \epsilon(m) \geq \frac{1}{n},$$

where the last inequality holds by choosing $\epsilon(m)$ small enough. \triangleleft

Theorem 15 follows from Claims 13 and 14. \blacktriangleleft

4.2 Pseudorandom Generators from Non-Uniform Hardness

In the non-uniform setting, it is not hard to construct a pseudorandom generator from average-case hardness.

► **Theorem 15.** *Let $\epsilon(n) := n^{-c}$ for some constant $c > 0$. Let \mathfrak{C} be a complexity class such that $\mathsf{P}^\mathfrak{C} = \mathsf{P}$. If $\mathfrak{C} \times \{\mathcal{U}\} \not\subseteq \text{Heur}_\epsilon \mathsf{P}/\text{poly}$, then, for any constant $\gamma > 1$, there exists a \mathfrak{C} -computable pseudorandom generator*

$$G = \left\{ G_n : \{0,1\}^n \rightarrow \{0,1\}^{n^\gamma} \right\}_{n \in \mathbb{N}}$$

secure against P/poly infinitely often.

Proof. This can be proved by a standard construction of a complexity-theoretic pseudorandom generator from an average-case hard function. Specifically, we combine the Nisan–Wigderson pseudorandom generator [36] with Yao’s XOR lemma [11].

Let $f = \{f_n : \{0,1\}^n \rightarrow \{0,1\}\}$ be a family of functions computable in \mathfrak{C} such that f_n cannot be computed by any polynomial-size circuit on a $(1 - \epsilon(n))$ fraction of inputs of length n . For a parameter k chosen later, let $f^{\oplus k}$ be a function such that $f^{\oplus k}(x_1, \dots, x_k) := f(x_1) \oplus \dots \oplus f(x_k)$. By Yao’s XOR lemma [11], there exists a polynomial $k \leq \text{poly}(n, \epsilon(n)^{-1})$ such that if f cannot be computed by a polynomial-size circuit on a $(1 - \epsilon(n))$ fraction of

inputs of length n , then for every constant $d \in \mathbb{N}$, $f^{\oplus k}$ cannot be computed by a polynomial-size circuit on a $(\frac{1}{2} + n^{-d})$ fraction of inputs of length n . For every constant $\gamma > 1$, Nisan and Wigderson [36] presented a pseudorandom generator construction

$$\text{NW}^{f^{\oplus k}} = \left\{ \text{NW}_n : \{0, 1\}^n \rightarrow \{0, 1\}^{n^\gamma} \right\}_{n \in \mathbb{N}}$$

based on an average-case hard function $f^{\oplus k}$ such that if for every constant $d \in \mathbb{N}$, $f^{\oplus k}$ cannot be computed by a polynomial-size circuit on a $(\frac{1}{2} + n^{-d})$ fraction of inputs of length n , then $\text{NW}^{f^{\oplus k}}$ is secure against P/poly . \blacktriangleleft

Applying Theorem 15 to $\mathfrak{C} := \text{PH}$, we obtain the following corollary.

► **Corollary 16.** *If $\text{PH} \times \{\mathcal{U}\} \not\subseteq \text{HeurP/poly}$, then there exists a PH -computable pseudorandom generator secure against P/poly .*

Proof. Lemma 4 shows that $\text{PH} \times \{\mathcal{U}\} \not\subseteq \text{HeurP/poly}$ if and only if $\text{PH} \times \{\mathcal{U}\} \not\subseteq \text{Heur}_\epsilon \text{P/poly}$, where $\epsilon(n) := n^{-1}$. The result now follows from Theorem 15. \blacktriangleleft

4.3 Putting It Together

We now exclude the non-uniform version of PH Pessiland.

► **Theorem 17** (Excluding PH Pessiland – the non-uniform case). *For every function $t: \mathbb{N} \rightarrow \mathbb{N}$ such that $n^\epsilon \leq t(n) \leq n^{1/\epsilon}$ for all large $n \in \mathbb{N}$, where $\epsilon > 0$ is an arbitrary constant, the following are equivalent.*

1. $\text{PH} \times \{\mathcal{U}\} \not\subseteq \text{HeurP/poly}$.
2. $\text{DistPH} \not\subseteq \text{HeurP/poly}$.
3. $\text{MINK}^{t, \text{PH}}[n - \log n] \times \{\mathcal{U}\} \not\subseteq \text{HeurP/poly}$.
4. *There exists a PH -computable pseudorandom generator*
 $G = \{G_n: \{0, 1\}^n \rightarrow \{0, 1\}^{n+1}\}_{n \in \mathbb{N}}$ *secure against P/poly infinitely often.*
5. *For every constant c , there exists a PH -computable pseudorandom generator*

$$G = \left\{ G_n: \{0, 1\}^n \rightarrow \{0, 1\}^{n^c} \right\}_{n \in \mathbb{N}}$$

secure against P/poly infinitely often.

Proof.

- (Item 1 \Rightarrow 5) This is Corollary 16.
- (Item 5 \Rightarrow 4) This follows from Lemma 10.
- (Item 4 \Rightarrow 3) This is Corollary 9.
- (Item 3 \Rightarrow 1) This immediately follows from the fact that $\text{MINKT}^{\text{PH}} \subseteq \text{PH}$.
- (Item 1 \Leftrightarrow 2) This follows from Lemma 4. \blacktriangleleft

5 Excluding Uniform PH Pessiland

In this section, we exclude the uniform version of PH Pessiland.

5.1 Uniform Hardness from Pseudorandom Generators

We now move on to excluding the uniform version of PH Pessiland. In this subsection, we prove that PH is hard on average if there is a PH-computable pseudorandom generator. The key is the following lemma, whose proof is based on [3, 39].

► **Lemma 18.** *If $\text{DistPH} \subseteq \text{HeurBPP}$, then $\text{PH} \times \text{PSamp}^{\text{PH}} \subseteq \text{HeurBPP}$.*

Proof. Consider a distributional problem $(L, \mathcal{D}) \in \text{PH} \times \text{PSamp}^{\text{PH}}$. We prove that $(L, \mathcal{D}) \in \text{HeurBPP}$.

We first show that \mathcal{D} can be “approximated” by a polynomial-time algorithm. Let S be a PH-computable sampler for \mathcal{D} ; that is, for some polynomial p , for every $n \in \mathbb{N}$, the distribution of $S(1^n, r)$ over $r \sim \{0, 1\}^{p(n)}$ is identical to \mathcal{D}_n . We may assume without loss of generality that the output length of $S(1^n, \cdot)$ is less than $p(n)$. Consider a decision version L_S of S defined as $L_S := \{(1^n, r, i, b) \mid S(1^n, r)_i = b \in \{1, \perp\}\}$, where x_i denotes the i -th bit of a binary string x if $i \leq |x|$ and \perp otherwise. Since S can be computed in PH, we have $L_S \in \text{PH}$. Let \mathcal{D}_S be the family of distributions $\{\mathcal{D}_{S,n}\}_{n \in \mathbb{N}}$ such that $\mathcal{D}_{S,n}$ is the distribution of $(1^n, r, i, b)$ over $r \sim \{0, 1\}^{p(n)}$, $i \sim [p(n)]$ and $b \sim \{1, \perp\}$. Observe that $(L_S, \mathcal{D}_S) \in \text{DistPH} \subseteq \text{HeurBPP}$. Therefore, there exists a randomized polynomial-time algorithm S_1 such that for every $n \in \mathbb{N}$ and every $\delta^{-1} \in \mathbb{N}$,

$$\Pr_{(1^n, r, i, b) \sim \mathcal{D}_{S,n}, S_1} [S_1(1^n, r, i, b; n, \delta) \neq L_S(1^n, r, i, b)] \leq \delta.$$

Let $S_2(r; n, \delta) := (S_1(1^n, r, 1, 1; n, \delta'), \dots, S_1(1^n, r, m, 1; n, \delta'))$, where $\delta' := \delta/2p(n)$ and m is the least index $i \in [p(n)]$ such that $S_1(1^n, r, i+1, \perp; n, \delta') = 1$. Observe that if $S_1(1^n, r, i, b; n, \delta') = L_S(1^n, r, i, b)$ for every $i \in [p(n)]$ and every $b \in \{1, \perp\}$, then $S_2(r; n, \delta) = S(1^n, r)$. Thus, by a union bound, we obtain

$$\Pr_{r \sim \{0, 1\}^{p(n)}, S_2} [S_2(r; n, \delta) \neq S(1^n, r)] \leq \delta$$

for every $n \in \mathbb{N}$. This means that the distribution $\mathcal{D} \in \text{PSamp}^{\text{PH}}$ can be approximated by a randomized polynomial-time algorithm S_2 .

Let \mathcal{D}' denote the family of distributions $\{S_2(\mathcal{U}_{p(n)}; n, \delta)\}_{(n, \delta^{-1}) \in \mathbb{N}^2}$. Since S_2 is a randomized polynomial-time algorithm, we have $\mathcal{D}' \in \text{PSamp}$. By assumption, we obtain $(L, \mathcal{D}') \in \text{DistPH} \subseteq \text{HeurBPP}$. Let A be a randomized heuristic scheme for (L, \mathcal{D}') . Then, for every $(n, \delta^{-1}) \in \mathbb{N}^2$ and every $\epsilon^{-1} \in \mathbb{N}$, we have

$$\Pr_{x \sim \mathcal{D}'_{(n, \delta^{-1})}, A} [A(x; \langle n, \delta^{-1} \rangle, \epsilon) \neq L(x)] \leq \epsilon.$$

This is equivalent to

$$\Pr_{r \sim \{0, 1\}^{p(n)}, S_2, A} [A(S_2(r; n, \delta); \langle n, \delta^{-1} \rangle, \epsilon) \neq L(S_2(r; n, \delta))] \leq \epsilon.$$

Let B be an algorithm such that $B(x; n, \delta) := A(x; \langle n, 2\delta^{-1} \rangle, \delta/2)$. Then, for every $n \in \mathbb{N}$ and $\delta^{-1} \in \mathbb{N}$, we obtain

$$\begin{aligned} & \Pr_{x \sim \mathcal{D}_n, B} [B(x; n, \delta) \neq L(x)] \\ & \leq \Pr_{r \sim \{0, 1\}^{p(n)}, S_2} [S_2(r; n, \delta/2) \neq S(1^n, r)] + \Pr_{r \sim \{0, 1\}^{p(n)}, B, S_2} [B(S_2(r; n, \delta/2); n, \delta) \neq L(S_2(r; n, \delta/2))] \\ & \leq \delta/2 + \Pr_{r \sim \{0, 1\}^{p(n)}, A, S_2} [A(S_2(r; n, \delta/2); \langle n, 2\delta^{-1} \rangle, \delta/2) \neq L(S_2(r; n, \delta/2))] \\ & \leq \delta, \end{aligned}$$

which implies that $(L, \mathcal{D}) \in \text{HeurBPP}$. ◀

► **Corollary 19.** *If $\text{PH} \times \{\mathcal{U}\} \subseteq \text{HeurBPP}$, then there exists no PH -computable pseudorandom generator*

$$G = \{G_n : \{0,1\}^n \rightarrow \{0,1\}^{n+c \log n}\}_{n \in \mathbb{N}}$$

secure against BPP infinitely often.

Proof. Consider the family \mathcal{D} of distributions $\{\mathcal{D}_n\}_{n \in \mathbb{N}}$ such that \mathcal{D}_n is the distribution of $G_n(\mathcal{U}_n)$ with probability $\frac{1}{2}$ and $\mathcal{U}_{n+c \log n}$ with probability $\frac{1}{2}$. Let $L := \{G_{|z|}(z) \mid z \in \{0,1\}^*\} \in \text{PH}$. Since $\mathcal{D} \in \text{PSamp}^{\text{PH}}$, it follows from Lemmas 4 and 18 that there exists a randomized heuristic scheme A for (L, \mathcal{D}) such that for every $n \in \mathbb{N}$ and every $\delta^{-1} \in \mathbb{N}$,

$$\Pr_{x \sim \mathcal{D}_n, A} [A(x; n, \delta) \neq L(x)] \leq \delta.$$

Fix an integer $n \in \mathbb{N}$. Below, for $\delta := \frac{1}{8}$, we claim that $A(\cdot; n, \delta)$ is a distinguisher for G_n . For simplicity, we write $A(\cdot)$ instead of $A(\cdot; n, \delta)$. By the definition of L and \mathcal{D} , we obtain

$$\frac{1}{2} \cdot \Pr_{z \sim \{0,1\}^n, A} [A(G_n(z)) \neq 1] + \frac{1}{2} \cdot \Pr_{x \sim \{0,1\}^{n+c \log n}, A} [A(x) \neq L(x)] \leq \delta.$$

Therefore, we have

$$\Pr_{z \sim \{0,1\}^n, A} [A(G_n(z)) = 1] \geq 1 - 2\delta$$

and

$$\Pr_{x \sim \{0,1\}^{n+c \log n}, A} [A(x) = 1] \leq \Pr_{x, A} [x \in L] + \Pr_{x, A} [A(x) \neq L(x)] \leq n^{-c} + 2\delta.$$

Thus, we conclude that

$$\Pr_{z \sim \{0,1\}^n, A} [A(G_n(z)) = 1] - \Pr_{x \sim \{0,1\}^{n+c \log n}, A} [A(x) = 1] \geq 1 - 4\delta - n^{-c} \geq \frac{1}{4}. \quad \blacktriangleleft$$

5.2 Pseudorandom Generators from Uniform Hardness

Trevisan and Vadhan [41] introduced a notion of advice (denoted by “//”) that can depend on the internal randomness of a randomized algorithm. Here, we apply the notion to an error-prone randomized algorithm.

► **Definition 20.** *For a function $\delta: \mathbb{N} \rightarrow [0, 1]$ and a distributional problem (L, \mathcal{D}) , we say that $(L, \mathcal{D}) \in \text{Heur}_\delta \text{BPP} // \log$ if there exist a polynomial p , a deterministic polynomial-time algorithm A , and an advice function $\alpha: \{0,1\}^* \rightarrow \{0,1\}^*$ such that $|\alpha(r)| \leq O(\log |r|)$ for every $r \in \{0,1\}^*$ and for any $n \in \mathbb{N}$,*

$$\Pr_{r \sim \{0,1\}^{p(n)}} \left[\Pr_{x \sim \mathcal{D}_n} [A(x, r, \alpha(r)) \neq L(x)] \leq \delta(n) \right] \geq \frac{1}{2}.$$

► **Fact 21.** *The success probability $\frac{1}{2}$ in Definition 20 can be amplified to $1 - 2^{-p(n)}$ for every polynomial p .*

Proof Sketch. Let A be the deterministic polynomial-time algorithm and α be the advice function as in Definition 20. Define an algorithm A' that takes random bits $r_1, \dots, r_{p(n)}$ and an advice string (a, i) as input and outputs $A(x, r_i, a)$. With probability at least $1 - 2^{-p(n)}$

over a random choice of $r_1, \dots, r_{p(n)}$, there exists some $i \in [p(n)]$ such that $A(\cdot, r_i, \alpha(r_i))$ and $L(\cdot)$ are close to each other. We define a new advice function α' that indicates such an index $i \in [p(n)]$ as well as $\alpha(r_i)$. Then, A' satisfies the property that

$$\Pr_{\bar{r}} \left[\Pr_{x \sim \mathcal{D}_n} [A'(x, \bar{r}, \alpha'(\bar{r})) \neq L(x)] \leq \delta(n) \right] \geq 1 - 2^{-p(n)},$$

where $\bar{r} = (r_1, \dots, r_{p(n)})$. ◀

We show that a logarithmic amount of advice can be removed when computing DistPH.

► **Theorem 22.** *If $\text{DistPH} \subseteq \text{Heur}_{n-c}\text{BPP}/\log$ for every constant $c > 0$, then $\text{DistPH} \subseteq \text{HeurBPP}$.*

Proof. By induction on k , we prove that $\text{Dist}\Sigma_k^P \subseteq \text{Heur}_{n-c'}\text{BPP}$ for every $k \in \mathbb{N}$ and every constant c' . (Note that this is sufficient because of Lemma 4.) The claim is obvious when $k = 0$ because $\Sigma_0^P = P$. Consider $k \geq 1$. Let $(L, \mathcal{D}) \in \text{Dist}\Sigma_k^P$. Since $\Sigma_k^P = \exists \cdot \Pi_{k-1}^P$, there exists a language $L' \in \Pi_{k-1}^P$ such that for some polynomial p , for every $x \in \{0, 1\}^*$,

- if $x \in L$, then $(x, y) \in L'$ for some $y \in \{0, 1\}^{p(|x|)}$ and
- if $x \notin L$, then $(x, y) \notin L'$ for every $y \in \{0, 1\}^{p(|x|)}$.

We use the search-to-decision reduction of [43, 3] to find a certificate $y \in \{0, 1\}^{p(|x|)}$. Specifically, Valiant and Vazirani [43] showed that there exists a randomized nonadaptive oracle polynomial-time algorithm R that takes x as input and oracle access to some problem $L_1 \in \Sigma_k^P$ and outputs a list $Y \subseteq \{0, 1\}^{p(|x|)}$ of strings such that if $x \in L$ then $(x, y) \in L'$ for some $y \in Y$ with probability at least $1 - 2^{-|x|^d}$ over a coin flip sequence of R , where d is an arbitrary constant. (We emphasize that it may be infeasible to check whether $(x, y) \in L'$ because L' is not necessarily in P . By allowing the reduction R to output a list of candidate certificates, the success probability of R can be amplified by repetition and can be assumed to be exponentially close to 1.)

Let c be an arbitrary large constant such that the running time of R is at most $n^{c/2}$. Let Q be the query distribution of R . Since $(L_1, Q \circ \mathcal{D}) \in \text{Dist}\Sigma_k^P \subseteq \text{Heur}_{n-c}\text{BPP}/\log$, there exist a polynomial p' , an advice function $\alpha_1 : \{0, 1\}^* \rightarrow \mathbb{N}$ and a deterministic polynomial-time algorithm A_1 such that for every $n \in \mathbb{N}$,

1. $\alpha_1(r) \in [p'(n)]$ for every $r \in \{0, 1\}^{p'(n)}$ and
2. with probability at least $1 - 2^{-n}$ over $r \sim \{0, 1\}^{p'(n)}$, it holds that

$$\Pr_{q \sim Q \circ \mathcal{D}_n} [A_1(q, r, \alpha_1(r)) \neq L_1(q)] \leq n^{-c}.$$

Let A be a randomized algorithm that takes (x, r, a) as input and simulates $R^{A_1(\cdot, r, a)}(x)$ to compute a list Y of candidate certificates and outputs Y . Since $R^{L_1}(x)$ outputs a correct list with high probability, by a union bound, with probability at least $1 - 2^{-n+1}$ over $r \sim \{0, 1\}^{p'(n)}$ and a coin flip sequence of R , it holds that

$$\Pr_{x \sim \mathcal{D}_n} [L'(x, y) \neq L(x) \text{ for every } y \in A(x, r, \alpha_1(r))] \leq n^{-c} \cdot n^{c/2} \leq n^{-c/2}. \quad (7)$$

Consider a family \mathcal{D}' of distributions $\{\mathcal{D}'_n\}_{n \in \mathbb{N}}$ such that \mathcal{D}'_n is the distribution of (x, y) such that $x \sim \mathcal{D}_n$, $r \sim \{0, 1\}^{p(n)}$, $a \sim [p'(n)]$, $Y := A(x, r, a)$, and $y \sim Y$. By induction hypothesis, we obtain $(L', \mathcal{D}') \in \text{Dist}\Pi_{k-1}^P \subseteq \text{Heur}_\delta\text{BPP}$, where $\delta(n) := n^{-c}/p'(n)$. Let B be a randomized heuristic algorithm for (L', \mathcal{D}') .

We are ready to describe a randomized heuristic algorithm C that witnesses $(L, \mathcal{D}) \in \text{Heur}_{n-c'} \text{BPP}$. The algorithm $C(x; n)$ picks $r \sim \{0, 1\}^{p'(n)}$ and outputs 1 if and only if $B(x, y; n) = 1$ for some y such that y is in the list produced by $A(x, r, a)$ for some $a \in [p'(n)]$.

We prove the correctness of C . Specifically, for every $n \in \mathbb{N}$, we claim that

$$\Pr_{x \sim \mathcal{D}_n, C} [C(x; n) \neq L(x)] \leq n^{-c'}.$$

By the definition of B , we have

$$\Pr_{(x, y) \sim \mathcal{D}'_n, B} [B(x, y; n) \neq L'(x, y)] \leq \delta(n).$$

Using the definition of \mathcal{D}'_n and a union bound, we obtain

$$\begin{aligned} & \Pr_{x \sim \mathcal{D}_n, r \sim \{0, 1\}^{p'(n)}, B} [B(x, y; n) \neq L'(x, y) \text{ for some } y \in A(x, r, a) \text{ and some } a \in [p'(n)]] \\ & \leq \delta(n) \cdot p'(n) \cdot n^{c/2} \leq n^{-c/2}. \end{aligned}$$

Combining this with Equation (7), with probability at least $1 - 2n^{-c/2} - 2^{-n+1}$ over $x \sim \mathcal{D}_n$, $r \sim \{0, 1\}^{p'(n)}$, and the internal randomness of B , we have $B(x, y; n) = L'(x, y)$ for every $y \in A(x, r, a)$ and every $a \in [p'(n)]$ and $L'(x, y) = L(x)$ for some $y \in A(x, r, a)$. Under this event, if $x \in L$, then there exist some $a := \alpha_1(r)$ and some $y \in A(x, r, a)$ such that $B(x, y; n) = L'(x, y) = 1$; thus, C accepts. If $x \notin L$, then for every a and every $y \in A(x, r, a)$, we have $B(x, y; n) = L'(x, y) = 0$; thus, C rejects. Therefore, the success probability of C is at least $1 - 2n^{-c/2} - 2^{-n+1}$, which is at least $1 - n^{-c'}$ by choosing $c := 3c'$. \blacktriangleleft

Using the powerful notion of “//” advice, we present a pseudorandom generator construction based on a uniform hardness assumption.

► **Lemma 23.** *Let $c > 0$ be a constant. If there exists no PH-computable pseudorandom generator*

$$G = \{G_n : \{0, 1\}^n \rightarrow \{0, 1\}^{n+c \log n}\}_{n \in \mathbb{N}}$$

secure against BPP infinitely often, then $\text{PH} \times \{\mathcal{U}\} \subseteq \text{Heur}_{n-d} \text{BPP}/\log$ for every constant d .

Proof. Let $f \in \text{PH}$. Fix an input length $n \in \mathbb{N}$. Let $t := n^{d+1}$. Let $k \in \mathbb{N}$ be a parameter such that $c \log(tnk) \leq k$ and $k = O(\log n)$. Let $f^{\oplus t}(x_1, \dots, x_t) = f(x_1) \oplus \dots \oplus f(x_t)$. We define $G : \{0, 1\}^{tnk} \rightarrow \{0, 1\}^{tnk+k}$ to be a “ k -wise direct product generator” [17]: Specifically, let

$$G(z_1, \dots, z_k) := (z_1, \dots, z_k, f^{\oplus t}(z_1), \dots, f^{\oplus t}(z_k)),$$

where $z_i \in \{0, 1\}^{tn}$ for each $i \in [k]$.

Since G maps a seed of length tnk to a string of length $tnk + k \geq tnk + c \log(tnk)$, by assumption, G is not secure against BPP infinitely often. That is, there exists a randomized polynomial-time algorithm A such that for some polynomial p , for all large $n \in \mathbb{N}$,

$$\left| \Pr_{w, A} [A(w) = 1] - \Pr_{\bar{z}, A} [A(G(\bar{z})) = 1] \right| > \frac{1}{p(n)}, \quad (8)$$

where $w \sim \{0, 1\}^{tnk+k}$ and $\bar{z} \sim \{0, 1\}^{tnk}$. Using a standard hybrid argument as in [36, 42] (see also [17]), there exists a deterministic polynomial-time algorithm A' such that, for some advice function α , it holds that

$$\Pr_{r, z} [A'(z, r, \alpha(r)) = f^{\oplus t}(z)] \geq \frac{1}{2} + \frac{1}{k \cdot p(n)},$$

where $z \sim \{0,1\}^{tn}$ and the length of an advice string $\alpha(r)$ is at most $k+1$ (which is used to specify whether the value inside the absolute function in Equation (8) is positive or negative and the bits $f^{\oplus t}(z_1), \dots, f^{\oplus t}(z_k)$). Impagliazzo, Jaiswal, Kabanets, and Wigderson [25] presented an almost uniform version of Yao's XOR lemma [11] that uses $O(\log kp(n))$ bits of advice: They showed that A' can be converted into an algorithm B such that with high probability over a random choice of r' , it holds that

$$\Pr_{x \sim \{0,1\}^n} [B(x, r', \alpha'(r')) = f(x)] \geq 1 - \delta,$$

where $\delta = O(\log(k \cdot p(n))/t) \leq n^{-d}$ and $|\alpha'(r)| \leq k+1 + O(\log kp(n))$. It follows that $(f, \mathcal{U}) \in \text{Heur}_{n-d}\text{BPP}/\log$. \blacktriangleleft

We observe that the k -wise direct product generator used in Lemma 23 is entropy-preserving in the sense of [33]. This observation leads to the DistPH-hardness of MINKT^{PH}:

► Lemma 24. *Let $\tau(n) = n^{\Omega(1)}$ be a function. If $\text{MINK}^{\tau, \text{PH}}[n - \log n] \times \{\mathcal{U}\} \subseteq \text{HeurBPP}$, then $\text{PH} \times \{\mathcal{U}\} \subseteq \text{Heur}_{n-d}\text{BPP}/\log$ for every constant d .*

Proof. We consider the same settings as Lemma 23: Let $f \in \text{PH}$ and fix an input length $n \in \mathbb{N}$. Let $t := n^{d+1}$. Let $k = O(\log n)$ be a parameter chosen later. We define $G: \{0,1\}^{tnk} \rightarrow \{0,1\}^{tnk+k}$ such that

$$G(z_1, \dots, z_k) := (z_1, \dots, z_k, f^{\oplus t}(z_1), \dots, f^{\oplus t}(z_k)),$$

where $z_i \in \{0,1\}^{tn}$ for each $i \in [k]$. It is sufficient to prove that there exists a randomized polynomial-time algorithm A such that for all large $n \in \mathbb{N}$,

$$\Pr_{z,A} [A(G(z)) = 1] - \Pr_{w,A} [A(w) = 1] \geq \frac{1}{2},$$

where $w \sim \{0,1\}^{tnk+k}$ and $z \sim \{0,1\}^{tnk}$. (Indeed, the proof of Lemma 23 shows that such an algorithm A can be converted into an algorithm that witnesses $(f, \mathcal{U}) \in \text{Heur}_{n-d}\text{BPP}/\log$.)

Let $L := \text{MINK}^{\tau, f}[n - \log n]$. Let B be a randomized heuristic scheme for (L, \mathcal{U}) . Since f can be computed in polynomial time given oracle access to the oracle $f \in \text{PH}$, we have $K^{n^{O(1)}, f}(G(z)) \leq tnk + O(\log n)$ for every $z \in \{0,1\}^{tnk}$. We use a padding argument to reduce the time bound: By choosing a constant c large enough so that $n^{O(1)} \ll \tau(n^c)$, we have $K^{\tau(m), f}(G(z) \cdot w) \leq tnk + n^c + O(\log n)$ for every $z \in \{0,1\}^{tnk}$ and every $w \in \{0,1\}^{n^c}$, where $m := tnk + k + n^c$. This is because each bit of $G(z) \cdot w$ can be efficiently computed using random access to a description (z, w, t, n, k, c) . We also have

$$K^{\tau(m), f}(G(z) \cdot w) \leq tnk + n^c + O(\log n) \leq m - \log m$$

by choosing $k = O(\log n)$ large enough. This implies that $G(z) \cdot w \in L$.

Let \mathcal{D} denote the distribution of $G(z) \cdot w$ such that $z \sim \{0,1\}^{tnk}$ and $w \sim \{0,1\}^{n^c}$. For every $y \in \text{supp}(\mathcal{D})$, we have $\mathcal{D}(y) \leq 2^{-m}2^k = 2^{-m}n^{O(1)}$; thus, \mathcal{D} is dominated by \mathcal{U}_m . Using the argument of domination, for every $\delta^{-1} \in \mathbb{N}$, we obtain

$$\Pr_{y \sim \mathcal{D}, B} [B(y; m, \delta) \neq L(y)] \leq \Pr_B [B(\mathcal{U}_m; m, \delta) \neq L(\mathcal{U}_m)] \cdot n^{O(1)} \leq \delta \cdot n^{O(1)}.$$

Therefore, we have

$$\Pr_{\substack{z \sim \{0,1\}^{tnk} \\ w \sim \{0,1\}^{n^c}, B}} [B(G(z) \cdot w; m, \delta) = 1] = \Pr_{y \sim \mathcal{D}, B} [B(y; m, \delta) = L(y)] \geq 1 - \delta \cdot n^{O(1)}.$$

Now, consider a uniform distribution:

$$\Pr_B[B(\mathcal{U}_m; m, \delta) = 1] \leq \Pr_B[\mathcal{U}_m \in L] + \Pr_B[B(\mathcal{U}_m; m, \delta) \neq L(\mathcal{U}_m)] \leq \frac{2}{m} + \delta.$$

Let A be a randomized algorithm that takes $\omega \in \{0, 1\}^{tnk+k}$ as input, picks $w \sim \{0, 1\}^{n^c}$ randomly, and outputs $B(\omega \cdot w; m, \delta)$, where $\delta^{-1} = n^{O(1)}$ is a sufficiently large polynomial. Then, the two inequalities above imply that

$$\Pr_A[A(G(\mathcal{U}_{tnk})) = 1] - \Pr_A[A(\mathcal{U}_{tnk+k}) = 1] \geq \frac{1}{2}$$

as desired. \blacktriangleleft

While we focused on solving PH with respect to the uniform distribution, this does not lose the generality:

► **Lemma 25.** *If $\text{PH} \times \{\mathcal{U}\} \subseteq \text{Heur}_{n^{-c}}\text{BPP}/\log$ for every constant c , then $\text{DistPH} \subseteq \text{Heur}_{n^{-c}}\text{BPP}/\log$ for every constant c .*

Proof Sketch. This follows from the work of Impagliazzo and Levin [26] (see also Lemma 4), which shows that the uniform distribution is a hardest distribution for NP (as well as PH). We sketch a proof below. Under the assumption, it can be shown that there exists no one-way function: Indeed, the assumption that $\text{NP} \times \{\mathcal{U}\} \subseteq \text{Heur}_{n^{-c}}\text{BPP}/\log$ implies that there are polynomially many polynomial-time algorithms one of which is guaranteed to invert a given function (each algorithm is assigned one advice string of length $O(\log n)$). Whether a one-way function is successfully inverted or not can be verified in polynomial time; thus, we obtain a (single) polynomial-time algorithm that inverts any one-way function. Impagliazzo and Luby [27] showed that if there is no one-way function, then there is no distributional one-way function. Impagliazzo and Levin [26] (see also [6]) showed that using an inverter for a distributional one-way function, every distributional problem $(L, \mathcal{D}) \in \text{DistPH}$ reduces to $(L', \mathcal{U}) \in \text{PH} \times \{\mathcal{U}\} \subseteq \text{Heur}_{n^{-c}}\text{BPP}/\log$. \blacktriangleleft

Finally, we show that the non-uniform and uniform complexities of PH are equivalent if every tally language in PH is easy.

► **Lemma 26.** *If every tally language in PH is in BPP , then for every language $L \in \text{PH}$, it holds that $(L, \mathcal{U}) \in \text{HeurP/poly}$ implies $(L, \mathcal{U}) \in \text{HeurBPP}$.*

Proof. The idea is to find the lexicographically first circuit that computes the distributional problem (L, \mathcal{U}) , which can be formulated as a tally language in PH .

Using approximate counting [40], there exists a language $A \in \text{PH}$ such that for every string c that encodes an n -input circuit C and for every parameter $\delta^{-1} \in \mathbb{N}$,

- $\Pr_{x \sim \mathcal{U}_n}[C(x) \neq L(x)] \leq \delta/4$ implies that $(c, \delta^{-1}) \in A$, and
- $\Pr_{x \sim \mathcal{U}_n}[C(x) \neq L(x)] \geq \delta/2$ implies that $(c, \delta^{-1}) \notin A$.

Let p be some large polynomial. For every $(n, \delta^{-1}) \in \mathbb{N}^2$, let $c_{n,\delta}$ be the lexicographically first string c such that c encodes a circuit C of size $p(n/\delta)$ such that $(C, \delta^{-1}) \in A$. By choosing p large enough, such a string c is guaranteed to exist because $(L, \mathcal{U}) \in \text{HeurP/poly}$.

Consider a tally language $T \subseteq \{1\}^*$ defined as follows: $1^N \in T$ if and only if $N = \langle n, \delta^{-1}, i \rangle$ for some $(n, \delta^{-1}, i) \in \mathbb{N}^3$ and the i -th bit of the description of $c_{n,\delta}$ is 1. (For simplicity, we assume that a circuit is encoded by a prefix-free encoding.) Since $A \in \text{PH}$, we also have $T \in \text{PH}$. By the assumption that every tally language in PH is in BPP , we obtain that $T \in \text{BPP}$; let M_T be the BPP algorithm that computes T .

Now we present a HeurBPP algorithm M for (L, \mathcal{U}) . On input $(x; n, \delta)$, M first computes $c_{n,\delta}$ by running M_T on inputs $1^{\langle n, \delta^{-1}, i \rangle}$ for all i , computes a circuit C represented by $c_{n,\delta}$, and outputs $C(x)$. We may assume that the algorithm M successfully computes $c_{n,\delta}$ with probability $1 - \delta/2$. By the property of A , the probability that $C(x) \neq L(x)$ over $x \sim \mathcal{U}_n$ is at most $\delta/2$. Overall, the error probability of M is at most δ . \blacktriangleleft

5.3 Putting It Together

We now exclude the uniform version of PH Pessiland.

► **Theorem 27** (Excluding PH Pessiland – the uniform case). *For every function $t: \mathbb{N} \rightarrow \mathbb{N}$ such that $n^\epsilon \leq t(n) \leq n^{1/\epsilon}$ for all large $n \in \mathbb{N}$, where $\epsilon > 0$ is an arbitrary constant, the following are equivalent.*

1. $\text{PH} \times \{\mathcal{U}\} \not\subseteq \text{HeurBPP}$.
2. $\text{MINK}^{t, \text{PH}}[n - \log n] \times \{\mathcal{U}\} \not\subseteq \text{HeurBPP}$.
3. $\text{DistPH} \not\subseteq \text{HeurBPP}$.
4. $\text{PH} \times \text{PSamp}^{\text{PH}} \not\subseteq \text{HeurBPP}$.
5. $\text{PH} \times \{\mathcal{U}\} \not\subseteq \text{Heur}_{n-c}\text{BPP}/\log$ for some constant c .
6. There exists a PH-computable pseudorandom generator
 $G = \{G_n: \{0, 1\}^n \rightarrow \{0, 1\}^{n+1}\}_{n \in \mathbb{N}}$ secure against BPP infinitely often.
7. For every constant c , there exists a PH-computable pseudorandom generator

$$G = \{G_n: \{0, 1\}^n \rightarrow \{0, 1\}^{n+c \log n}\}_{n \in \mathbb{N}}$$

secure against BPP infinitely often.

8. Either for every constant $c > 1$, there exists a PH-computable pseudorandom generator

$$G = \{G_n: \{0, 1\}^n \rightarrow \{0, 1\}^{n^c}\}_{n \in \mathbb{N}}$$

secure against BPP infinitely often, or there exists a tally language L in PH such that $L \notin \text{BPP}$.

Proof.

(Item 1 \Rightarrow 3) (Item 3 \Rightarrow 4) These are evident from the definitions.

(Item 4 \Rightarrow 5) This follows by combining Theorem 22 and Lemmas 18 and 25.

(Item 5 \Rightarrow 7) This is Lemma 23.

(Item 7 \Rightarrow 6) Given a pseudorandom generator $G_n: \{0, 1\}^n \rightarrow \{0, 1\}^{n+c \log n}$, we define

$$G'_n: \{0, 1\}^n \rightarrow \{0, 1\}^{n+1}$$

so that $G'_n(z)$ is the first $n + 1$ bits of $G_n(z)$ for every $z \in \{0, 1\}^n$. It is easy to see that $G' = \{G'_n\}_{n \in \mathbb{N}}$ is a PH-computable pseudorandom generator secure against BPP infinitely often.

(Item 6 \Rightarrow 1) This is Corollary 19.

(Item 5 \Rightarrow 2) This is Lemma 24.

(Item 2 \Rightarrow 1) This follows from the fact that $\text{MINK}^{t, \text{PH}}[n - \log n] \subseteq \text{NP}^{\text{PH}} = \text{PH}$.

(Item 8 \Rightarrow 4) If there exists a PH-computable pseudorandom generator, then $\text{PH} \times \{\mathcal{U}\} \not\subseteq \text{HeurBPP}$ by Corollary 19. Otherwise, there exists a tally language L in $\text{PH} \setminus \text{BPP}$, in which case (L, \mathcal{T}) is a distributional problem in $\text{PH} \times \text{PSamp}$ but not in HeurBPP. Here, \mathcal{T} is the family of distributions $\{\mathcal{T}_n\}_{n \in \mathbb{N}}$ such that \mathcal{T}_n is the singleton distribution on 1^n .

(Item 4 \Rightarrow 8) We prove the contrapositive. The assumption that there exists no PH-computable pseudorandom generator implies that $\text{PH} \times \{\mathcal{U}\} \subseteq \text{HeurP/poly}$ by Corollary 16. Using Lemma 26, we obtain that $\text{PH} \times \{\mathcal{U}\} \subseteq \text{HeurBPP}$. \blacktriangleleft

References

- 1 Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. On basing one-way functions on NP-hardness. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 701–710, 2006. doi:[10.1145/1132516.1132614](https://doi.org/10.1145/1132516.1132614).
- 2 László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP Has Subexponential Time Simulations Unless EXPTIME has Publishable Proofs. *Computational Complexity*, 3:307–318, 1993. doi:[10.1007/BF01275486](https://doi.org/10.1007/BF01275486).
- 3 Shai Ben-David, Benny Chor, Oded Goldreich, and Michael Luby. On the Theory of Average Case Complexity. *J. Comput. Syst. Sci.*, 44(2):193–219, 1992. doi:[10.1016/0022-0000\(92\)90019-F](https://doi.org/10.1016/0022-0000(92)90019-F).
- 4 Manuel Blum and Silvio Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM J. Comput.*, 13(4):850–864, 1984. doi:[10.1137/0213053](https://doi.org/10.1137/0213053).
- 5 Andrej Bogdanov and Christina Brzuska. On Basing Size-Verifiable One-Way Functions on NP-Hardness. In *Proceedings of the Theory of Cryptography Conference (TCC)*, pages 1–6, 2015. doi:[10.1007/978-3-662-46494-6_1](https://doi.org/10.1007/978-3-662-46494-6_1).
- 6 Andrej Bogdanov and Luca Trevisan. Average-Case Complexity. *Foundations and Trends in Theoretical Computer Science*, 2(1), 2006. doi:[10.1561/0400000004](https://doi.org/10.1561/0400000004).
- 7 Andrej Bogdanov and Luca Trevisan. On Worst-Case to Average-Case Reductions for NP Problems. *SIAM J. Comput.*, 36(4):1119–1159, 2006. doi:[10.1137/S0097539705446974](https://doi.org/10.1137/S0097539705446974).
- 8 Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning Algorithms from Natural Proofs. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 10:1–10:24, 2016. doi:[10.4230/LIPIcs.CCC.2016.10](https://doi.org/10.4230/LIPIcs.CCC.2016.10).
- 9 Larry Carter and Mark N. Wegman. Universal Classes of Hash Functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979. doi:[10.1016/0022-0000\(79\)90044-8](https://doi.org/10.1016/0022-0000(79)90044-8).
- 10 Joan Feigenbaum and Lance Fortnow. Random-Self-Reducibility of Complete Sets. *SIAM J. Comput.*, 22(5):994–1005, 1993. doi:[10.1137/0222061](https://doi.org/10.1137/0222061).
- 11 Oded Goldreich, Noam Nisan, and Avi Wigderson. On Yao’s XOR-Lemma. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 273–301. Springer, 2011. doi:[10.1007/978-3-642-22670-0_23](https://doi.org/10.1007/978-3-642-22670-0_23).
- 12 Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A Pseudorandom Generator from any One-way Function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. doi:[10.1137/S0097539793244708](https://doi.org/10.1137/S0097539793244708).
- 13 Shuichi Hirahara. Identifying an Honest EXP^{NP} Oracle Among Many. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 244–263, 2015. doi:[10.4230/LIPIcs.CCC.2015.244](https://doi.org/10.4230/LIPIcs.CCC.2015.244).
- 14 Shuichi Hirahara. Non-black-box Worst-case to Average-case Reductions within NP. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 247–258, 2018.
- 15 Shuichi Hirahara. Characterizing Average-Case Complexity of PH by Worst-Case Meta-Complexity. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 50–60, 2020.
- 16 Shuichi Hirahara. Non-Disjoint Promise Problems from Meta-Computational View of Pseudorandom Generator Constructions. In *Proceedings of the Computational Complexity Conference (CCC)*, pages 20:1–20:47, 2020. doi:[10.4230/LIPIcs.CCC.2020.20](https://doi.org/10.4230/LIPIcs.CCC.2020.20).
- 17 Shuichi Hirahara. Unexpected hardness results for Kolmogorov complexity under uniform reductions. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 1038–1051, 2020. doi:[10.1145/3357713.3384251](https://doi.org/10.1145/3357713.3384251).
- 18 Shuichi Hirahara. Average-Case Hardness of NP from Exponential Worst-Case Hardness Assumptions. *Electron. Colloquium Comput. Complex.*, 28:58, 2021. To appear in STOC 2021. URL: <https://eccc.weizmann.ac.il/report/2021/058>.

- 19 Shuichi Hirahara and Rahul Santhanam. On the Average-Case Complexity of MCSP and Its Variants. In *Proceedings of the Computational Complexity Conference (CCC)*, pages 7:1–7:20, 2017. doi:10.4230/LIPIcs.CCC.2017.7.
- 20 Rahul Ilango. Approaching MCSP from Above and Below: Hardness for a Conditional Variant and $\text{AC}^0[p]$. In *Proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*, pages 34:1–34:26, 2020. doi:10.4230/LIPIcs.ITCS.2020.34.
- 21 Rahul Ilango. Constant Depth Formula and Partial Function Versions of MCSP are Hard. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 424–433, 2020.
- 22 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)*, pages 22:1–22:36, 2020. doi:10.4230/LIPIcs.CCC.2020.22.
- 23 Russell Impagliazzo. A Personal View of Average-Case Complexity. In *Proceedings of the Structure in Complexity Theory Conference*, pages 134–147, 1995. doi:10.1109/SCT.1995.514853.
- 24 Russell Impagliazzo. Hard-Core Distributions for Somewhat Hard Problems. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 538–545, 1995. doi:10.1109/SFCS.1995.492584.
- 25 Russell Impagliazzo, Ragesh Jaiswal, Valentine Kabanets, and Avi Wigderson. Uniform Direct Product Theorems: Simplified, Optimized, and Derandomized. *SIAM J. Comput.*, 39(4):1637–1665, 2010. doi:10.1137/080734030.
- 26 Russell Impagliazzo and Leonid A. Levin. No Better Ways to Generate Hard NP Instances than Picking Uniformly at Random. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 812–821, 1990. doi:10.1109/FSCS.1990.89604.
- 27 Russell Impagliazzo and Michael Luby. One-way Functions are Essential for Complexity Based Cryptography (Extended Abstract). In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 230–235, 1989. doi:10.1109/SFCS.1989.63483.
- 28 Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E Requires Exponential Circuits: Derandomizing the XOR Lemma. In *Proceedings of the Symposium on the Theory of Computing (STOC)*, pages 220–229, 1997. doi:10.1145/258533.258590.
- 29 Russell Impagliazzo and Avi Wigderson. Randomness vs Time: Derandomization under a Uniform Assumption. *J. Comput. Syst. Sci.*, 63(4):672–688, 2001. doi:10.1006/jcss.2001.1780.
- 30 Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 73–79, 2000. doi:10.1145/335305.335314.
- 31 Richard M. Karp and Richard J. Lipton. Turing machines that take advice. *L'Enseignement Mathématique*, 28:191–209, 1982.
- 32 Ker-I Ko. On the Complexity of Learning Minimum Time-Bounded Turing Machines. *SIAM J. Comput.*, 20(5):962–986, 1991. doi:10.1137/0220059.
- 33 Yanyi Liu and Rafael Pass. On One-way Functions and Kolmogorov Complexity. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 1243–1254, 2020.
- 34 Yanyi Liu and Rafael Pass. On One-way Functions from NP-Complete Problems. *Electron. Colloquium Comput. Complex.*, 28:59, 2021. URL: <https://eccc.weizmann.ac.il/report/2021/059>.
- 35 Andrei A. Muchnik and Nikolai K. Vereshchagin. Shannon Entropy vs. Kolmogorov Complexity. In *Computer Science - Theory and Applications, First International Computer Science Symposium in Russia, CSR 2006, St. Petersburg, Russia, June 8-12, 2006, Proceedings*, pages 281–291, 2006. doi:10.1007/11753728_29.
- 36 Noam Nisan and Avi Wigderson. Hardness vs Randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994. doi:10.1016/S0022-0000(05)80043-1.

- 37 Aduri Pavan, Rahul Santhanam, and N. V. Vinodchandran. Some Results on Average-Case Hardness Within the Polynomial Hierarchy. In *Proceedings of the Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 188–199, 2006. doi:[10.1007/11944836_19](https://doi.org/10.1007/11944836_19).
- 38 Rahul Santhanam. Pseudorandomness and the Minimum Circuit Size Problem. In *Proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*, pages 68:1–68:26, 2020. doi:[10.4230/LIPIcs.ITCS.2020.68](https://doi.org/10.4230/LIPIcs.ITCS.2020.68).
- 39 Rainer Schuler and Osamu Watanabe. Towards Average-Case Complexity Analysis of NP Optimization Problems. In *Proceedings of the Structure in Complexity Theory Conference*, pages 148–159, 1995. doi:[10.1109/SCT.1995.514854](https://doi.org/10.1109/SCT.1995.514854).
- 40 Larry J. Stockmeyer. The Complexity of Approximate Counting (Preliminary Version). In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 118–126, 1983. doi:[10.1145/800061.808740](https://doi.org/10.1145/800061.808740).
- 41 Luca Trevisan and Salil P. Vadhan. Pseudorandomness and Average-Case Complexity Via Uniform Reductions. *Computational Complexity*, 16(4):331–364, 2007. doi:[10.1007/s00037-007-0233-x](https://doi.org/10.1007/s00037-007-0233-x).
- 42 Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012. doi:[10.1561/0400000010](https://doi.org/10.1561/0400000010).
- 43 Leslie G. Valiant and Vijay V. Vazirani. NP is as Easy as Detecting Unique Solutions. *Theor. Comput. Sci.*, 47(3):85–93, 1986. doi:[10.1016/0304-3975\(86\)90135-0](https://doi.org/10.1016/0304-3975(86)90135-0).
- 44 Emanuele Viola. On Constructing Parallel Pseudorandom Generators from One-Way Functions. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 183–197, 2005. doi:[10.1109/CCC.2005.16](https://doi.org/10.1109/CCC.2005.16).
- 45 Emanuele Viola. The complexity of constructing pseudorandom generators from hard functions. *Computational Complexity*, 13(3-4):147–188, 2005. doi:[10.1007/s00037-004-0187-1](https://doi.org/10.1007/s00037-004-0187-1).
- 46 Emanuele Viola. The Sum of D Small-Bias Generators Fools Polynomials of Degree D . *Computational Complexity*, 18(2):209–217, 2009. doi:[10.1007/s00037-009-0273-5](https://doi.org/10.1007/s00037-009-0273-5).
- 47 Andrew Chi-Chih Yao. Theory and Applications of Trapdoor Functions (Extended Abstract). In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 80–91, 1982. doi:[10.1109/SFCS.1982.45](https://doi.org/10.1109/SFCS.1982.45).