Electronic Colloquium on Computational Complexity, Report No. 138 (2018)

Non-black-box Worst-case to Average-case Reductions within NP

Shuichi Hirahara*
The University of Tokyo

August 10, 2018

Abstract

There are significant obstacles to establishing an equivalence between the worst-case and average-case hardness of NP: Several results suggest that black-box worst-case to average-case reductions are not likely to be used for reducing any worst-case problem outside coNP to a distributional NP problem.

This paper overcomes the barrier. We present the first *non-black-box* worst-case to average-case reduction from a problem outside coNP (unless Random 3SAT is easy for coNP algorithms) to a distributional NP problem. Specifically, we consider the minimum time-bounded Kolmogorov complexity problem (MINKT), and prove that there exists a zero-error randomized polynomial-time algorithm approximating the minimum time-bounded Kolmogorov complexity k within an *additive* error $\widetilde{O}(\sqrt{k})$ if its average-case version admits an errorless heuristic polynomial-time algorithm. (The converse direction also holds under a plausible derandomization assumption.) We also show that, given a truth table of size 2^n, approximating the minimum circuit size within a factor of $2^{(1-\epsilon)n}$ is in BPP for some constant $\epsilon > 0$ if and only if its average-case version is easy.

Based on our results, we propose a research program for excluding Heuristica, i.e., establishing an equivalence between the worst-case and average-case hardness of NP through the lens of MINKT or the Minimum Circuit Size Problem (MCSP).

1 Introduction

The main result of this paper is to establish a relationship between two long-standing open questions in complexity theory.

Theorem (informal)**.** *If an approximation version of* MINKT *or* MCSP *is* NP*-hard, then Heuristica does not exist, that is, the average-case and worst-case hardness of* NP *are equivalent.*

Based on this, we propose resolving the former question as a potentially feasible research program towards excluding Heuristica. We elaborate on the two open questions below.

1.1 Impagliazzo's Five Worlds

Impagliazzo [31] gave an influential survey on average-case complexity, and explored five possible worlds: Algorithmica (where NP is easy on the worst-case; e.g. $\mathsf{P} = \mathsf{NP}$), Heuristica (where NP is

*hirahara@is.s.u-tokyo.ac.jp

ISSN 1433-8092

hard on the worst-case, but easy on the average-case; e.g. $\mathsf{P} \neq \mathsf{NP}$ and $\mathsf{DistNP} \subseteq \mathsf{AvgP}$), Pessiland (where NP is hard on average, but there is no one-way function), Minicrypt (where a one-way function exists, but no public-key cryptography exists), and Cryptomania (public-key cryptography exists). These are classified according to four central open questions in complexity theory, and exactly one of the worlds corresponds to our world.

What is known about Impagliazzo's five worlds? The list of the five worlds is known to be in "decreasing order" of the power of polynomial-time machines; that is, $\exists$ public-key cryptography $\Rightarrow$ $\exists$ one-way functions $\Rightarrow$ $\mathsf{DistNP} \not\subseteq \mathsf{AvgP} \Rightarrow \mathsf{P} \neq \mathsf{NP}$. The converse directions of these implications are important open questions in complexity theory; that is, True $\overset{?}{\Rightarrow}$ $\mathsf{P} \neq \mathsf{NP} \overset{?}{\Rightarrow} \mathsf{DistNP} \not\subseteq \mathsf{AvgP}$ $\overset{?}{\Rightarrow}$ $\exists$ one-way functions $\overset{?}{\Rightarrow}$ $\exists$ public-key cryptography. By establishing one implication, one possible world is excluded from Impagliazzo's five worlds. And if the four implications are proved, it is concluded that our world is Cryptomania, i.e., computationally-secure public-key cryptography exists.

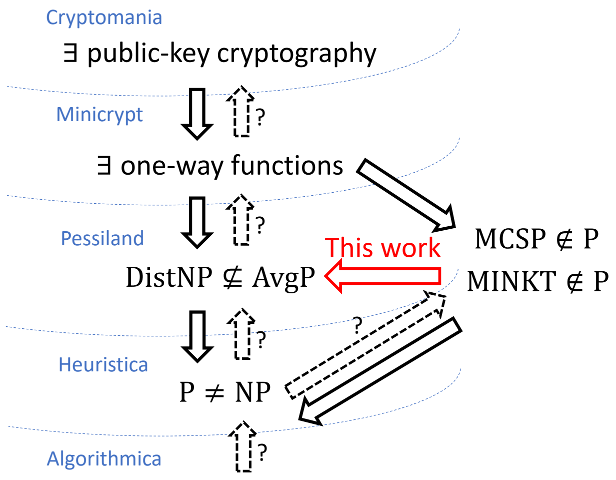

Figure 1: Impagliazzo's five worlds. Note that this figure ignores details such as the difference between P and BPP; MCSP and its approximation version GapMCSP.

1.2 Minimum Circuit Size Problem (MCSP) and Its Variants

Another long-standing open question in complexity theory, whose importance is best explained with Impagliazzo's five worlds, is the complexity of MCSP, or its Kolmogorov complexity variants, such as MKTP or MINKT. The *Minimum Circuit Size Problem* (MCSP [36]) asks, given a function $f\colon \{0,1\}^n \to \{0,1\}$ represented as its entire truth table of size 2^n together with an integer $s \in \mathbb{N}$, whether there exists a circuit of size at most s computing f. Similarly, MINKT (Minimum Kolmogorov Time-bounded Complexity [38]) asks the minimum *program size* to output a given string x within a given time bound t; specifically, given a string x and integers t, s represented in unary, it asks whether there is a program of size $\leq s$ that outputs x within t steps. (There is another variant called MKTP [5, 10], which aims at minimizing $s + t$, i.e., the program size plus the time it takes to output x by a random access machine.)

These problems are easily shown to be in NP. However, no NP-completeness proof has been found, nor no evidence against NP-completeness (under weak reducibility notions) has been found so far. This is despite the fact that MCSP is recognized as a fundamental problem as early as 1950s in the Soviet Union [49]. Indeed, it is reported in [11] that Levin delayed his publication on the NP-completeness of SAT [40] because he wanted to prove a similar result for MCSP. It is thus a long-standing open problem in complexity theory whether MCSP is NP-complete or not. The open question is depicted in Figure 1 as the implication "$\mathsf{NP} \neq \mathsf{P} \overset{?}{\Rightarrow} \mathrm{MCSP} \notin \mathsf{P}$."[1]

A fundamental relationship between cryptography and MCSP was discovered in the celebrated natural proof framework of Razborov and Rudich [46], based on which Kabanets and Cai [36] reawakened interest in MCSP. Since then many efforts have been made to understand the complexity of MCSP (e.g., [5, 8, 11, 6, 42, 30, 28, 17, 44, 27, 9, 7, 33, 26]). In particular, any one-way function can be inverted if MCSP (or MINKT) is in BPP (cf. [22, 24, 5]). This corresponds to the implication "$\exists$ one-way functions $\Rightarrow$ MCSP $\notin \mathsf{BPP}$."

This paper shows that if an approximation version of MCSP or MINKT cannot be solved in BPP, then its average-case version is not in AvgP. In particular, NP-completeness of the approximation problem excludes Heuristica, i.e., a world where $\mathsf{NP} \not\subseteq \mathsf{BPP}$ and $\mathsf{DistNP} \subseteq \mathsf{AvgP}$. The latter is a central open question in the theory of average-case complexity, as we review next.

1.3 Average-case Complexity

A traditional complexity class such as P and NP measures the performance of an algorithm with respect to the *worst-case* input. However, such a worst-case input may not be found efficiently, and may never be encountered in practice. Average-case complexity, pioneered by Levin [39], aims at analyzing the performance of an algorithm with respect to *random inputs* which can be easily generated by an efficient algorithm.

Specifically, a *distributional problem* $(L, \mathcal{D})$ is a pair of a language $L \subseteq \{0,1\}^*$ and a family of distributions $\mathcal{D} = \{\mathcal{D}_m\}_{m\in\mathbb{N}}$. A family of distributions $\mathcal{D}$ is said to be *efficiently samplable* if there exists a randomized polynomial-time algorithm that, given an integer $m \in \mathbb{N}$ represented in unary, outputs a string distributed according to $\mathcal{D}_m$. DistNP is the class of distributional problems $(L, \mathcal{D})$ such that $L \in \mathsf{NP}$ and $\mathcal{D}$ is efficiently samplable. The performance of an algorithm for a distributional problem $(L, \mathcal{D})$ is measured by the average-case behavior of A on input chosen according to $\mathcal{D}_m$, for each $m \in \mathbb{N}$; specifically, for a failure probability $\delta \colon \mathbb{N} \to [0,1]$, $\mathsf{Avg}_\delta\mathsf{P}$ denotes the class of distributional problems $(L, \mathcal{D})$ that admit an *errorless heuristic polynomial-time algorithm* A; that is, $A(x)$ outputs the correct answer $L(x)$ or otherwise a special failure symbol $\perp$ for every input x, and $A(x)$ outputs $\perp$ with probability at most $\delta(m)$ over the random choice of $x \sim \mathcal{D}_m$, for every instance size $m \in \mathbb{N}$. We define $\mathsf{AvgP} := \bigcap_{c\in\mathbb{N}} \mathsf{Avg}_{m^{-c}}\mathsf{P}$. The reader is referred to the survey of Bogdanov and Trevisan [14] for detailed background on average-case complexity.

The central open question in this area is whether Heuristica exists. That is, does worst-case hardness on NP such as $\mathsf{NP} \not\subseteq \mathsf{BPP}$ imply $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$? Worst-case to average-case reductions are known for complexity classes much higher than NP, or specific problems in $\mathsf{NP}\cap\mathsf{coNP}$: For complexity classes above the polynomial-time hierarchy such as PSPACE and EXP, a general technique based on error-correcting codes provides a worst-case to average-case reduction (cf. [21, 12, 48]).

Problems based on lattices admit worst-case to average-case reductions from some problems in $\mathsf{NP}\cap\mathsf{coNP}$ to distributional NP problems. In a seminal paper of Ajtai [3], it is shown that an approx-

[1] Note that a problem L is NP-hard under polynomial-time Turing reductions iff $\mathsf{NP} \not\subseteq \mathsf{P}^R \Rightarrow L \notin \mathsf{P}^R$ for every oracle R. The unrelativized implication $\mathsf{NP} \not\subseteq \mathsf{P} \Rightarrow L \notin \mathsf{P}$ gives rise to the weakest notion of NP-hardness.

imation version of the shortest vector problem of a lattice in $\mathbb{R}^n$ admits a worst-case to average-case reduction. The complexity of approximating the length of a shortest vector depends greatly on an approximation factor. A worst-case to average-case reduction is known when an approximation factor is larger than $\widetilde{O}(n)$ [41]. Note that Heuristica does not exist if this approximation problem is NP-hard; however, this is unlikely because approximating the length of a shortest vector within a factor of $O(\sqrt{n})$ is in $\mathsf{NP} \cap \mathsf{coNP}$ [2]. Some NP-hardness is known for an approximation factor of $n^{O(1/\log\log n)}$ [25].

1.4 Barriers for Worst-case to Average-case Reductions within NP

There are significant obstacles to establishing worst-case to average-case connections for NP-complete problems (e.g., [21, 15, 53, 4, 32, 29]). A standard technique to establish worst-case to average-case connections is by "black-box" reductions, meaning that a hypothetical heuristic algorithm is regarded as a (possibly inefficient) oracle. Building on Feigenbaum and Fortnow [21], Bogdanov and Trevisan [15] showed that if a language L reduces to a distributional NP problem via a black-box nonadaptive randomized polynomial-time reduction, then $L \in \mathsf{NP/poly} \cap \mathsf{coNP/poly}$. Here, the advice "$\mathsf{/poly}$" is mainly used to encode some information about the distributional problem, and can be removed in some cases such as a reduction to inverting one-way functions [4, 13] or breaking hitting set generators [29]. Therefore, in order to reduce any problem outside $\mathsf{NP} \cap \mathsf{coNP}$ to a distributional NP problem, it is likely that a non-black-box reduction technique is needed.[2]

Gutfreund, Shaltiel and Ta-Shma [23] developed a non-black-box technique to show a worst-case to "average-case" reduction; however, the notion of "average-case" is different from the usual one. They showed that, under the assumption that $\mathsf{P} \neq \mathsf{NP}$, for every polynomial-time algorithm A trying to compute SAT, there exists an efficiently samplable distribution $\mathcal{D}_A$ under which A fails to compute SAT on average. The hard distribution $\mathcal{D}_A$ depends on a source code of A, and hence it is not necessarily true that there exists a fixed distribution under which SAT is hard on average.

In contrast, we consider the following two simple distributions. One is the uniform distribution, denoted by $\mathcal{U}$, under which an instance x of size m is generated by choosing $x \in_R \{0,1\}^m$ uniformly at random. The other is a uniform distribution with auxiliary unary input, denoted by $\mathcal{D}^u$, under which an instance $(x, 1^t)$ of size m is generated by choosing an integer $t \in_R \{1, \dots, m\}$ and a string $x \in_R \{0,1\}^{m-t}$ uniformly at random.

1.5 Our Results

The main contribution of this paper is to present the first *non-black-box* worst-case to average-case reduction from a problem conjectured to be outside $\mathsf{NP} \cap \mathsf{coNP}$ to a distributional NP problem.

Recall the notion of time-bounded Kolmogorov complexity: For a string $x \in \{0,1\}^*$, the *Kolmogorov complexity* $\mathrm{K}_t(x)$ of x within time t is defined as the length of a shortest program M such that M outputs x within t steps. For example, 0^n can be described as "output 0 n times," which can be encoded as a binary string of length $\log n + O(1)$; thus $\mathrm{K}_t(0^n) = \log n + O(1)$ for a sufficiently large t. Kolmogorov complexity enables us to define the notion of *randomness* for a finite string x. We say that a string $x \in \{0,1\}^*$ is *r-random* with respect to K_t if $\mathrm{K}_t(x) \geq r(|x|)$, for a function $r \colon \mathbb{N} \to \mathbb{N}$.

[2] Here we implicitly used a popular conjecture that $\mathsf{AM} = \mathsf{NP}$ [37], and ignored the possibility that an *adaptive* black-box reduction could be used to overcome the barriers.

Our main technical result is a search to average-case reduction between the following two problems. One is a search problem of approximating $\mathrm{K}_t(x)$ within an *additive* error term of $\widetilde{O}\big(\sqrt{\mathrm{K}_t(x)}\big)$ on input $(x, 1^t)$, where $\widetilde{O}$ hides some $\mathsf{polylog}(|x|)$ factor. The other is a distributional NP problem, denoted by $(\mathrm{MINKT}[r], \mathcal{D}^u)$, of deciding, on input $(x, 1^t)$ sampled from $\mathcal{D}^u$, whether x is not r-random with respect to K_t.

Theorem 1 (Main)**.** *Let $r\colon \mathbb{N} \to \mathbb{N}$ be any function such that for some constant $c > 0$, for all large $n \in \mathbb{N}$, $n - c\sqrt{n}\log n \le r(n) < n$. Assume that $(\mathrm{MINKT}[r], \mathcal{D}^u) \in \mathsf{Avg}_{1/6m}\mathsf{P}$. Then, for some function $\sigma(n, s) = s + O\big((\log n)\sqrt{s} + (\log n)^2\big)$ and some polynomial $\tau(n, t)$, there exists a zero-error randomized polynomial-time algorithm that, on input $(x, 1^t)$, outputs a program M of size $|M| \le \sigma(|x|, \mathrm{K}_t(x))$ such that M outputs x in $\tau(|x|, t)$ steps.*

There is a natural decision version associated with the search problem above, denoted by $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$. This is the promise problem of deciding, on input $(x, 1^t, 1^s)$, whether $\mathrm{K}_t(x) \le s$ or $\mathrm{K}_{t'}(x) > \sigma(|x|, s)$ for $t' = \tau(|x|, t)$. Using Theorem 1, we prove the following worst-case and average-case equivalence between the worst-case problem $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ and the distributional NP problem $(\mathrm{MINKT}[r], \mathcal{D}^u)$.

Corollary 2. *In the following list, we have $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 4$. Moreover, if* Promise-$\mathsf{ZPP}$ = Promise-P*, then we also have $4 \Rightarrow 2$.*

1. $\mathsf{DistNP} \subseteq \mathsf{AvgP}$.
2. $(\mathrm{MINKT}[r], \mathcal{D}^u) \in \mathsf{Avg}_{1/6m}\mathsf{P}$ *for some $r\colon \mathbb{N} \to \mathbb{N}$ such that $n - O\big(\sqrt{n}\log n\big) \le r(n) < n$ for all large $n \in \mathbb{N}$.*
3. *There exists a zero-error randomized polynomial-time algorithm solving the search version of* $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$*, for some $\sigma(n, s) = s + O\big((\log n)\sqrt{s} + (\log n)^2\big)$ and some polynomial $\tau(n, t)$.*
4. $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in$ Promise-ZPP *for some $\sigma(n, s) = s + O\big((\log n)\sqrt{s} + (\log n)^2\big)$ and some polynomial $\tau(n, t)$.*

Note that the derandomization hypothesis Promise-ZPP = Promise-P follows from the plausible circuit lower bound $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}(2^{\Omega(n)})$ [34].

We also establish similar results for MCSP. Specifically, we show that the complexity of the following two problems is the same with respect to BPP algorithms. One is a promise problem, denoted by $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ for a constant $\epsilon > 0$, of approximating the minimum circuit size within a factor of $2^{(1-\epsilon)n}$ on input the truth table of a function $f\colon \{0,1\}^n \to \{0,1\}$. The other is a distributional NP problem, denoted by $(\mathrm{MCSP}[2^{\epsilon n}], \mathcal{U})$ for a constant $\epsilon > 0$, of deciding whether the minimum circuit size is at most $2^{\epsilon n}$ given the truth table of a function $f\colon \{0,1\}^n \to \{0,1\}$ chosen uniformly at random.

Theorem 3 (see Theorem 50)**.** *The following are equivalent.*

1. $\mathrm{Gap}_\epsilon\mathrm{MCSP} \in$ Promise-BPP *for some $\epsilon > 0$.*
2. *There exists a randomized polynomial-time algorithm solving the search version of* $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ *for some $\epsilon > 0$.*
3. $(\mathrm{MCSP}[2^{\epsilon n}], \mathcal{U}) \in \mathsf{AvgBPP}$ *for some constant $\epsilon \in (0, 1)$.*

Previously, an equivalence between the worst-case and average-case complexity of MCSP with respect to "feasibly-on-average" algorithms (meaning that the error set of an algorithm is recognized by some efficient algorithm) was shown under the assumption that one-way functions exist [27]; however, the assumption is so strong that the equivalence becomes trivial when the feasibly-on-average algorithm itself is an efficient algorithm. Independently of our work, Igor C. Oliveira and Rahul Santhanam (personal communication) obtained a worst-case to average-case connection for a version of MCSP called MAveCSP, which asks if there exists a small circuit approximating a given function f.

1.6 Hardness of GapMINKT

We argue that our techniques are essentially non-black-box. If Theorem 1 were established via a nonadaptive black-box worst-case to average-case reduction, then by using the techniques of Bogdanov and Trevisan [15], we would obtain $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in \mathsf{coNP/poly}$. This is unlikely, as we discuss below. (In fact, our non-black-box reduction can be regarded as a nonadaptive reduction to breaking a hitting set generator; thus, the advice "/poly" is not indispensable [29].)

Unfortunately, basing hardness of MCSP or MINKT on worst-case hardness assumptions is a very challenging task. The best known worst-case hardness result for MCSP (which also holds for MINKT) is SZK (statistical zero knowledge) hardness, which is proved by inverting some auxiliary-input one-way function (Allender and Das [6]). This cannot be seen as evidence that $\mathrm{MCSP} \notin \mathsf{coNP}$ since $\mathsf{SZK} \subseteq \mathsf{AM} \cap \mathsf{coAM}$. There is evidence that the SZK-hardness is the best that one can hope for the current reduction techniques: A certain (one-query randomized) reduction technique called an *oracle-independent* reduction [28] cannot be used to base hardness of MCSP on any problem beyond $\mathsf{AM} \cap \mathsf{coAM}$. Here, a reduction to MCSP is said to be oracle-independent if the reduction can be generalized to a reduction to MCSP^A for every oracle A.

Fortunately, we can still argue hardness of MCSP or MINKT based on average-case assumptions. Indeed, MKTP is known to be Random 3SAT-hard [27], which provides evidence that $\mathrm{MKTP} \notin \mathsf{coNP}$. To prove similar average-case hardness results, we observe that, given $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ as oracle, one can break any hitting set generator.

Proposition 4. *Let σ, τ be the parameters as in Theorem 1. Any efficiently computable hitting set generator $H = \{H_n \colon \{0,1\}^n \to \{0,1\}^{n+\widetilde{\omega}(\sqrt{n})}\}_{n\in\mathbb{N}}$ is not secure against a polynomial-time algorithm with oracle access to $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$.*

This is because any range of a hitting set generator is not random in the sense of time-bounded Kolmogorov complexity; thus, to test whether x is in the range of H, it suffices to check whether $\mathrm{K}_t(x)$ is small.

One example of hitting set generators conjectured to be secure against nondeterministic algorithms comes from the natural proof framework. Rudich [47] conjectured that there is no $\mathsf{NP/poly}$-natural property useful against $\mathsf{P/poly}$. In particular, under his conjecture, we have $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \notin \mathsf{coNP/poly}$.

More importantly, Random 3SAT can be viewed as a hitting set generator (which extends its seed of length N by $\Omega(N/\log N)$ bits) that is conjectured to be secure against coNP algorithms. *Random 3SAT* is a widely investigated problem algorithmically (e.g., [18, 20, 19]). This is the problem of checking the satisfiability of a 3CNF formula randomly generated by choosing m clauses uniformly at random from all the possible clauses on n variables. The best coNP algorithm solving Random 3SAT on average is the algorithm given by Feige, Kim and Ofek [19], which works when

$m > O(n^{7/5})$; this is better than the best deterministic algorithm, which works when $m > O(n^{3/2})$ [20].

We show that if $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in \mathsf{coNP}$, there is a much better algorithm than [19]; specifically, for any constant $\Delta > 1/\log(8/7) \approx 5.19$ and for $m := \Delta n$, Random 3SAT with m clauses can be solved by an errorless coNP algorithm with probability $1 - 2^{-\Omega(n)}$. Ryan O'Donnell (cf. [27, 16]) conjectured that there is no coNP algorithm solving Random 3SAT with $m = \Delta n$ clauses for a sufficiently large constant Δ with high probability. Thus under his conjecture, we have $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \notin \mathsf{coNP}$.

1.7 Perspective: An Approach Towards Excluding Heuristica

We propose a research program towards excluding Heuristica through the lens of MCSP or MINKT. Note that if $\mathsf{NP} \leq_T^{\mathsf{BPP}} \mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ then we obtain the following by Theorem 1: If $\mathsf{NP} \not\subseteq \mathsf{BPP}$ then $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$, which means that Heuristica does not exist.

Unfortunately, there are still several obstacles we need to overcome in order for this research program to be completed. Although our proofs overcome the limits of black-box reductions, our proofs do *relativize*. And there is a relativization barrier for excluding Heuristica: Impagliazzo [32] constructed an oracle A such that $\mathsf{DistNP}^A \subseteq \mathsf{AvgP}^A$ and $\mathsf{NP}^A \cap \mathsf{coNP}^A \not\subseteq \mathsf{P}^A/\mathsf{poly}$. Under the same oracle, it follows from a relativized version of Theorem 1 that $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}^A$ is not NP^A-hard under $\mathsf{P}^A/\mathsf{poly}$-Turing reductions. Thus it requires some nonrelativizing technique to establish NP-hardness of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ even under P/poly-Turing reductions. (Previously, Ko [38] constructed a relativized world where MINKT is not NP-hard under P-Turing reductions.)

We also mention that there are a number of results (e.g. [36, 5, 10, 42, 28, 30, 9]) showing that proving NP-hardness (under reducibility notions stronger than P/poly-Turing reductions) of MCSP is extremely difficult or impossible. For example, Murray and Williams [42] showed that MCSP is provably not NP-hard under some sublinear time reductions; similarly, NP-hardness of GapMCSP under polynomial-time Turing reductions implies $\mathsf{EXP} \neq \mathsf{ZPP}$ [28], which is a notorious open question.

Now we conjecture that the following is a feasible research question.

Conjecture 5. *Let σ, τ be the parameters as in Theorem 1.* $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ *is* NP*-hard under* $\mathsf{coNP}/\mathsf{poly}$*-Turing reductions. That is,* $\mathsf{NP} \subseteq \mathsf{coNP}^A/\mathsf{poly}$ *for any oracle A solving* $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$.

Note that the choice of reducibility is somewhat subtle: The relativization barrier applies to P/poly reductions, but it is not known whether a similar barrier applies to $\mathsf{coNP}/\mathsf{poly}$ reductions. Ko [38] also speculated that MINKT might be NP-complete under $\mathsf{NP} \cap \mathsf{coNP}$ reductions. We mention that there is a nonrelativizing proof technique to prove PSPACE-completeness of a space-bounded version of MINKT (cf. [5]).

A positive answer to Conjecture 5 implies the following: If $\mathsf{NP} \not\subseteq \mathsf{coNP}/\mathsf{poly}$, then $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$. This will base the hardness of DistNP on a plausible worst-case assumption of NP, and in particular, an assumption that the polynomial-time hierarchy does not collapse. Currently, no worst-case hardness assumption on the polynomial-time hierarchy is known to imply $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$.

1.8 Our Techniques

At a high level, our contributions are to further explore the interplay between Kolmogorov-randomness and the hardness versus randomness framework. Allender, Buhrman, Koucký, van Melkebeek, Ronneburger [5] exploited the interplay and presented a number of results on the power of Kolmogorov-random strings: *Pseudorandom bits are not Kolmogorov-random*, and hence the set of Kolmogorov-random strings can be used to break pseudorandom generators, based on which they demonstrated the power of Kolmogorov-random strings. For this purpose, they used previously constructed pseudorandom generators in a black-box manner. In contrast, we open the black box and take a closer look at the interplay between Kolmogorov-randomness and pseudorandomness.

Specifically, our starting point is the Nisan-Wigderson generator [43]. They presented a (complexity-theoretic) pseudorandom generator NW^f secure against small circuits, based on any "hard" function f (in the sense that f cannot be approximated by small circuits, that is, $\Pr_x[f(x) = C(x)] \le \frac{1}{2} + \epsilon$ for some small $\epsilon > 0$ and any small circuit C).

Its security is proved by the following reduction: Given any statistical test T that distinguishes the output distribution of NW^f from the uniform distribution, one can construct a small T-oracle circuit C^T that approximates f. If T can be implemented by a small circuit, then this is a contradiction to the assumption that f is hard; thus the pseudorandom generator is secure. Such a security proof turns out to be quite fruitful not only for derandomization [37, 35, 51], but also for Trevisan's extractor [50], investigating the power of Kolmogorov-random strings [5], and the generic connection between learning and natural proof [17].

Our proofs also make use of a security proof. It enables us to transform any statistical test T for NW^f to a small circuit C^T that describes a $(\frac{1}{2} + \epsilon)$-fraction of the truth table of f. Moreover, as observed in [35], such small circuits can be constructed efficiently. By using a list-decodable error-correcting code Enc, given any statistical test T for $\mathrm{NW}^{\mathrm{Enc}(x)}$, one can efficiently find a short description for x under the oracle T.

We argue that there is a statistical test T for $\mathrm{NW}^{\mathrm{Enc}(x)}$ under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$. Consider the distributional NP problem $(\mathrm{MINKT}[r], \mathcal{D}^u)$. A crucial observation is that there are few nonrandom strings (i.e., compressible by a short program); that is, there are few Yes instances in $\mathrm{MINKT}[r]$. Thus any errorless heuristic algorithm solving $(\mathrm{MINKT}[r], \mathcal{D}^u)$ must reject a large fraction of random strings. This gives rise to a dense subset $T \in P$ of random strings, and it can be shown that T is a statistical test for any hitting set generator.

As a consequence, we obtain an efficient algorithm that, on input x, outputs a short program d describing x under the oracle T. Since T can be accepted by some polynomial-time algorithm (that comes from the errorless heuristic algorithm for $(\mathrm{MINKT}[r], \mathcal{D}^u)$), we can describe x by using the description d and *a source code* of the algorithm accepting T. This is the crucial part in which our proof is non-black-box; we need a source code of the errorless heuristic algorithm in order to have a short description for x. We then obtain a randomized polynomial-time search algorithm for $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$.

The proof sketch above enables us to find a somewhat short description, but it is not sufficient to obtain a description of length $(1 + o(1)) \cdot \mathrm{K}_t(x)$, nor to obtain the Random 3SAT-hardness of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$. To optimize the quality of the approximation, we need to exploit an improvement of the Nisan-Wigderson generator (and Trevisan's extractor), given by Raz, Reingold and Vadhan [45].

Finally, the randomized algorithm described above can be made zero-error; indeed, if $\mathsf{DistNP} \subseteq \mathsf{AvgZPP}$, then any randomized algorithm can be made zero-error (as mentioned in [31] without a proof). This is because a Kolmogorov-random string w can be found by picking a string uniformly at random, and one can check whether w is Kolmogorov-random or not by using an errorless heuristic algorithm for $(\mathrm{MINKT}[r], \mathcal{D}^u)$; by using w as a source of a hard function and invoking the hardness versus randomness framework again, we can derandomize the rest of the randomized computation. (The zero-error algorithm may fail only if no Kolmogorov-random string is found.)

Interestingly, we invoke the hardness versus randomness framework *twice* for completely different purposes. On one hand, to derandomize a randomized computation, it is desirable to minimize the seed length of a pseudorandom generator, because we need to exhaustively search all the seeds. On the other hand, to obtain a short description, it is desirable to minimize the *output* length of a pseudorandom generator (or, in other words, to maximize the seed length); this is because the efficiency of the security proof is dominated by the output length.

To prove a similar equivalence between worst-case and average-case hardness of MCSP, there is one difficulty: An error-correcting code Enc may significantly increase the circuit complexity of f. As a consequence, for a function f that can be computed by a small circuit, the circuit complexity of the output of $\mathrm{NW}^{\mathrm{Enc}(f)}$ is not necessarily small, and thus an errorless heuristic algorithm for MCSP may not induce a statistical test for $\mathrm{NW}^{\mathrm{Enc}(f)}$; here, the circuit complexity of a string x refers to the size of a smallest circuit whose truth table is x. Nevertheless, it is still possible to amplify the hardness of f while preserving the circuit complexity of f. Indeed, Carmosino, Impagliazzo, Kabanets, and Kolokolova [17] established a generic reduction from approximately learning to natural properties, by using the fact that a natural property is a statistical test for $\mathrm{NW}^{\mathrm{Amp}(f)}$, where $\mathrm{Amp}(f)$ denotes a hardness amplified version of f. We observe that their approximately learning is enough to achieve the approximation factor stated in Theorem 3. Moreover, as shown by Hirahara and Santhanam [27], a natural property is essentially an errorless heuristic algorithm for MCSP. By combining these results, we obtain a search to average-case reduction for GapMCSP.

Open Problems. In addition to the main open problem (Conjecture 5), there are several open problems unanswered in this paper. In Theorem 1, we assumed that there exists an errorless heuristic *deterministic* algorithm for $(\mathrm{MINKT}[r], \mathcal{D}^u)$; we do not know whether $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ is easy if $\mathsf{DistNP} \subseteq \mathsf{AvgZPP}$. A naive approach is to have a description that incorporates random bits of AvgZPP algorithms, but it spoils the quality of the approximation. Another open question is whether a similar non-black-box reduction is possible for HeurP, that is, a heuristic algorithm that may err. We crucially rely on the fact that there are few YES instances in $\mathrm{MINKT}[r]$, and hence our techniques do not seem to be easily extended to the case of a heuristic algorithm with error.

Organization. In Section 2, we review background on Kolmogorov complexity. Then in Section 3, we give a search to average-case reduction for MINKT, assuming the existence of some oracle; the existence of the oracle is justified in Section 4, which completes the proof of Theorem 1. In Section 5, we present evidence against $\mathrm{MINKT} \in \mathsf{coNP}$. Section 6 is devoted to proving Theorem 3.

2 Preliminaries

Notation. For an integer $n \in \mathbb{N}$, let $[n] := \{1, \dots, n\}$. For a language $A \subseteq \{0,1\}^*$ and an integer $n \in \mathbb{N}$, let $A^{=n} := A \cap \{0,1\}^n$.

For a finite set D, we indicate by $x \in_R D$ that x is picked uniformly at random from the set D. For a probability distribution $\mathcal{D}$, we indicate by $x \sim \mathcal{D}$ that x is a random sample from $\mathcal{D}$.

For a function $f \colon \{0,1\}^\ell \to \{0,1\}$, we denote by $\mathtt{tt}(f)$ its truth table, i.e., $f(z_1) \cdots f(z_{2^\ell})$ where $z_1, \dots, z_{2^\ell} \in \{0,1\}^\ell$ are all the strings of length ℓ in the lexicographic ordering. We will sometimes identify a function f and its truth table $\mathtt{tt}(f)$, and vice versa.

Language. A set $L \subseteq \{0,1\}^*$ of strings is called a *language*. We identify L with its characteristic function $L\colon \{0,1\}^* \to \{0,1\}$ such that $L(x) = 1$ iff $x \in L$ for every $x \in \{0,1\}^*$.

Promise Problem. A *promise problem* is a pair $(L_{\mathrm{Y}}, L_{\mathrm{N}})$ of languages $L_{\mathrm{Y}}, L_{\mathrm{N}} \subseteq \{0,1\}^*$ such that $L_{\mathrm{Y}} \cap L_{\mathrm{N}} = \varnothing$, where L_{Y} and L_{N} are regarded as the set of Yes and No instances, respectively. If $L_{\mathrm{Y}} = \{0,1\}^* \setminus L_{\mathrm{N}}$, we identify $(L_{\mathrm{Y}}, L_{\mathrm{N}})$ with the language $L_{\mathrm{Y}} \subseteq \{0,1\}^*$. We say that a language A *solves* a promise problem $(L_{\mathrm{Y}}, L_{\mathrm{N}})$ if $L_{\mathrm{Y}} \subseteq A \subseteq \{0,1\}^* \setminus L_{\mathrm{N}}$. For a complexity class $\mathfrak{C}$ such as ZPP and BPP, we denote by Promise-$\mathfrak{C}$ the promise version of $\mathfrak{C}$.

Circuits. For a Boolean circuit C, we denote by $|C|$ the size of circuit C; the measure of circuit size (e.g., the number of gates, wires or the description length) is not important for our results; for concreteness, we assume that the size is measured by the number of gates. We identify a circuit C on n variables with the function $C\colon \{0,1\}^n \to \{0,1\}$ computed by C. For a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, denote by $\mathsf{size}(f)$ the size of a minimum circuit C computing f.

Kolmogorov Complexity. We fix any efficient *universal Turing machine* U. This is a Turing machine that takes as input a description of any Turing machine M together with a string x, and simulates M on input x efficiently. We will only need the following fact.

Fact 6 (Universal Turing machine)**.** *There exists a polynomial p_U such that, for any machine M, there exists some description $d_M \in \{0,1\}^*$ of M such that, for every input $x \in \{0,1\}^*$, if $M(x)$ stops in t steps for some $t \in \mathbb{N}$ then $U(d_M, x)$ outputs $M(x)$ within $p_U(t)$ steps.*

For simplicity of notation, we identify M with its description d_M. We sometimes regard $p_U(t) = t$ for simplifying statements of claims. For a string x, its Kolmogorov complexity is the length of a shortest description for x. Formally:

Definition 7 (Time-bounded Kolmogorov complexity)**.** *For any oracle $A \subseteq \{0,1\}^*$, any string $x \in \{0,1\}^*$, and any integer $t \in \mathbb{N}$, the* Kolmogorov complexity *of x within time t relative to A is defined as* $\mathrm{K}_t^A(x) := \min\{\, |d| \mid U^A(d) = x \text{ in } t \text{ steps } \}$.

To explain a consequence of the security proof of the Nisan-Wigderson generator, it is convenient to introduce an approximation version of Kolmogorov complexity.

Definition 8 (Approximation version of Time-bounded Kolmogorov complexity)**.** *For functions $f, g\colon \{0,1\}^\ell \to \{0,1\}$, define* $\mathrm{dist}(f,g) := \Pr_{x \in_R \{0,1\}^\ell}[f(x) \neq g(x)]$. *For a function $f\colon \{0,1\}^\ell \to \{0,1\}$, an integer $t \in \mathbb{N}$, and an oracle $A \subseteq \{0,1\}^*$, define*

$$\mathrm{K}_{t,\delta}^A(f) := \min\{\, |d| \mid \ U^A(d) \text{ outputs } \mathtt{tt}(g) \text{ of length } 2^\ell \text{ within } t \text{ steps and } \mathrm{dist}(f,g) \le 1/2 - \delta \ \}.$$

Problems on Kolmogorov Complexity. MINKT is a problem asking for the time-bounded Kolmogorov complexity of x on input x and a time bound t.

Definition 9 (Ko [38])**.** *For any oracle $A \subseteq \{0,1\}^*$, define* $\mathrm{MINKT}^A := \{\, (x, 1^t, 1^s) \mid \mathrm{K}_t^A(x) \le s \,\}$.

It is easy to see that MINKT $\in \mathsf{NP}$, by guessing a certificate d of length at most s, and checking whether $U(d)$ outputs x within t steps. Such a certificate for MINKT will play a crucial role; thus we formalize it next.

Definition 10. *For an oracle $A \subseteq \{0,1\}^*$, integers $s, t \in \mathbb{N}$, and a string $x \in \{0,1\}^*$, a string $d \in \{0,1\}^*$ is called a* certificate for $\mathrm{K}_t^A(x) \preceq s$ *if $U^A(d)$ outputs x within t steps and $|d| \le s$. A certificate for $\mathrm{K}_{t,\delta}^A(x) \preceq s$ is defined in a similar way.*

In this terminology, for proving Theorem 1, on input $(x, 1^t)$, we seek a certificate for

$$\mathrm{K}_{t'}(x) \preceq \mathrm{K}_t(x) + O\big((\log|x|)\sqrt{\mathrm{K}_t(x)} + (\log|x|)^2\big)$$

for some $t' = \mathsf{poly}(|x|, t)$. Note here that "$\preceq$" is just a symbol, and "$\mathrm{K}_t(x) \preceq s$" should be interpreted as a tuple $(x, 1^t, 1^s)$, which is an instance of MINKT.

We also define a promise version of MINKT, parameterized by σ and τ.

Definition 11 (Promise version of MINKT)**.** *Let $\sigma, \tau \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be any functions such that $\sigma(n,s) \ge s$ and $\tau(n,t) \ge t$ for any $n, s, t \in \mathbb{N}$.* $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ *is a promise problem defined as follows.*

- Yes *instances: $(x, 1^t, 1^s)$ such that $\mathrm{K}_t(x) \le s$.*
- No *instances: $(x, 1^t, 1^s)$ such that $\mathrm{K}_{t'}(x) > \sigma(|x|, s)$ for $t' := \tau(|x|, t)$.*

When $\sigma(n,s) = s$ and $\tau(n,t) = t$, the promise problem $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ coincides with MINKT. It is also convenient to define the search version of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$.

Definition 12 (Search version of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$)**.** *For any functions σ, τ as in Definition 11, the search version of* $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ *is defined as follows.*

- *Inputs: A string $x \in \{0,1\}^*$ and an integer $t \in \mathbb{N}$ represented in unary.*
- *Output: A certificate for $\mathrm{K}_{t'}(x) \preceq \sigma(|x|, \mathrm{K}_t(x))$ for any $t' \ge \tau(|x|, t)$.*

A randomized algorithm A is called a zero-error *randomized algorithm solving the search version of* $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ *if, for every $x \in \{0,1\}^*$ and $t \in \mathbb{N}$, $A(x, 1^t)$ outputs a certificate for $\mathrm{K}_{t'}(x) \preceq \sigma(|x|, \mathrm{K}_t(x))$ whenever $A(x, 1^t) \ne \bot$, and $A(x, 1^t)$ outputs $\bot$ with probability at most $\frac{1}{2}$.*

We will show that, if every distributional NP can be solved by some errorless heuristic polynomial-time algorithm, then the search version of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ can be solved by a zero-error randomized polynomial-time algorithm for $\sigma(n,s) := s + O\big((\log n)\sqrt{s} + (\log n)^2\big)$ and some polynomial $\tau(n,t)$. As a corollary, we also obtain $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in \text{Promise-}\mathsf{ZPP}$ because of the following simple fact.

Fact 13 (Decision reduces to search)**.** *Let $\sigma, \tau \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be any efficiently computable and nondecreasing functions. If there exists a zero-error randomized polynomial-time algorithm solving the search version of* $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$*, then* $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in \text{Promise-}\mathsf{ZPP}$.

Proof. The main point is that the zero-error randomized search algorithm does not err in the sense that it outputs an approximately shortest certificate whenever it succeeds. Therefore, given a zero-error randomized algorithm M solving the search version of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$, the following algorithm solves the decision version: On input $(x, 1^t, 1^s)$, run M on input $(x, 1^t)$. If M outputs $\bot$, then output $\bot$ and halt. Otherwise, M outputs some certificate d. Accept iff $|d| \le \sigma(|x|, s)$ and $U(d)$ outputs x in $\tau(|x|, t)$ steps.

We claim the correctness of this algorithm. If $(x, 1^t, 1^s)$ is an YES instance of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$, then we obtain a certificate d for $\mathrm{K}_{t'}(x) \preceq \sigma(|x|, \mathrm{K}_t(x))$ where $t' := \tau(|x|, t)$ unless M outputs $\perp$; that is, $U(d)$ outputs x in t' steps, and $|d| \le \sigma(|x|, \mathrm{K}_t(x)) \le \sigma(|x|, s)$. Thus the algorithm above accepts with probability at least $\frac{1}{2}$. If $(x, 1^t, 1^s)$ is a NO instance of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$, then we have $\mathrm{K}_{t'}(x) > \sigma(|x|, s)$ for $t' := \tau(|x|, t)$. Thus the algorithm rejects unless M outputs $\perp$. □

The following is the crucial lemma in which our proof is non-black-box.

Lemma 14. *Let $T \in \mathsf{P}$. Then there exists some polynomial p such that $\mathrm{K}_{t'}(x) \le \mathrm{K}_t^T(x) + O(1)$ for any $x \in \{0,1\}^*$ and any t, t' such that $t' \ge p(t)$. Moreover, given a certificate for $\mathrm{K}_t^T(x) \preceq s$, one can efficiently find a certificate for $\mathrm{K}_{t'}(x) \preceq s + O(1)$.*

We will use this lemma for an errorless heuristic polynomial-time algorithm accepting T (in Theorem 1). Thus, the output of our non-black-box reduction will be a certificate for $\mathrm{K}_{t'}(x)$ which incorporates a source code of the errorless heuristic polynomial-time algorithm.

Proof. Let M_0 be a polynomial-time machine that accepts T. Consider the following machine M: On input $d \in \{0,1\}^*$, simulate $U^T(d)$ using M_0; that is, if U makes a query q to the oracle T, then run M_0 on input q and answer the query q with $M_0(q)$. Then M outputs what $U^T(d)$ outputs.

Now suppose that $U^T(d)$ outputs x in t steps. Then, by the definition, $M(d)$ outputs x in $p_0(t)$ steps for some polynomial p_0 (that depends only on the running time of M_0); thus, $U(M, d)$ outputs x in $p_U(p_0(t))$ steps, where p_U is the slowdown of the universal Turing machine. Hence we obtain $\mathrm{K}_{t'}(x) \le \mathrm{K}_t^T(x) + O(|M|)$ for $t' \ge p(t) := p_U(p_0(t))$.

To see the "moreover" part, given a certificate d for $\mathrm{K}_t^T(x) \preceq s$, we may simply output (M, d) as a certificate for $\mathrm{K}_{t'}(x) \preceq s + O(|M|)$. □

3 Search to Average-case Reductions for MINKT

In this section, we present an efficient algorithm that outputs a certificate for GapMINKT, given an oracle that accepts some *dense* subset of *random* strings. The existence of such an oracle will be justified in the next section under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$. We start with the definitions about an oracle. A string $x \in \{0,1\}^*$ is said to be *random* if x does not have a shorter description than itself. More generally:

Definition 15 (r-random)**.** *Let $r\colon \mathbb{N} \to \mathbb{N}$ be a function. We say that a string x is* r-random *with respect to K_t if $\mathrm{K}_t(x) \ge r(|x|)$. Let $R_t[r]$ denote the set of all r-random strings with respect to K_t.*

Definition 16 (dense)**.** *For every $m \in \mathbb{N}$ and $\delta \in [0,1]$, we say that a set $A \subseteq \{0,1\}^m$ of strings is* δ-dense *if $\Pr_{w \in_R \{0,1\}^m}[w \in A] \ge \delta$.*

In particular, a set $A \subseteq \{0,1\}^m$ is called a δ-dense subset of r-random strings $R_t[r]$ if $A \subseteq R_t[r]$ and $|A| \ge 2^m \delta$.

The main idea is that a dense subset of random strings gives rise to a statistical test distinguishing any pseudorandom generator from the uniform distribution. Indeed, take any efficiently computable function $G\colon \{0,1\}^d \to \{0,1\}^m$ where $d \lesssim r(m)$; then any range $G(z)$ of G can be described by its seed z in polynomial time; hence $G(z)$ is not r-random since $\mathrm{K}_t(G(z)) \lesssim d \lesssim r(m)$; thus a δ-dense subset T of r-random strings is a *statistical test* for G with advantage δ, i.e.,

$\left|\Pr_{w \in_R \{0,1\}^m}[w \in T] - \Pr_{z \in_R \{0,1\}^d}[G(z) \in T]\right| \ge \delta$. We will use this fact to break the Nisan-Wigderson generator.

We proceed to define the Nisan-Wigderson generator NW^f. Originally, Nisan and Wigderson [43] defined the notion of *design* as a family of subsets $S_1, \dots S_m$ such that $|S_i \cap S_j|$ is small for every distinct $i, j \in [m]$. As observed by Raz, Reingold and Vadhan [45], a weaker notion is sufficient for a security proof of the Nisan-Wigderson generator. Our notion is, however, different from the weak design defined in [45] due to some technical details.

Definition 17. *We say that a family $\mathcal{S} = (S_1, \dots, S_m)$ of subsets of $[d]$ is a (ℓ, ρ)-*design *if $|S_i| = \ell$ and $\sum_{j=1}^{i-1} 2^{|S_i \cap S_j|} + m - i \le \rho m$ for every $i \in [m]$.*

There is an efficient way to construct such a family with nice parameters.

Lemma 18 (follows from [45, Lemma 15])**.** *For any $m, \ell, d \in \mathbb{N}$ such that $d/\ell \in \mathbb{N}$, there exists a $(\ell, \exp(\ell^2/d))$-design $\mathcal{S}_{m,\ell,d} = (S_1, \dots, S_m) \subseteq \binom{[d]}{\ell}$. Moreover, the family $\mathcal{S}_{m,\ell,d}$ can be constructed by a deterministic algorithm in time $\mathsf{poly}(m, d)$.*

Proof Sketch. Raz, Reingold and Vadhan [45] showed how to construct, in time $\mathsf{poly}(m, d)$, a family of subsets $S_1, \dots, S_m \subseteq [d]$ of size ℓ such that $\sum_{j=1}^{i-1} 2^{|S_i \cap S_j|} \le (1 + \ell/d)^\ell \cdot (i-1) \le \exp(\ell^2/d) \cdot i$ for every $i \in [m]$. (The family is constructed by dividing $[d]$ into ℓ disjoint blocks of size d/ℓ, and, for each $i \in [m]$, choosing one random element out of each block and adding it to S_i. The construction can be derandomized by the method of conditional expectations.) The same family satisfies the condition that $\sum_{j=1}^{i-1} 2^{|S_i \cap S_j|} + m - i \le \exp(\ell^2/d) \cdot m$ for every $i \in [m]$. □

For a string $z \in \{0,1\}^d$ and a subset $S = \{i_1 < \cdots < i_\ell\} \subseteq [d]$, we denote by $z_S \in \{0,1\}^\ell$ the string $z_{i_1} \cdots z_{i_\ell}$. To avoid introducing a new variable, we treat d/ℓ as if it is a variable.

Definition 19 (Nisan-Wigderson generator [43])**.** *For a function $f \colon \{0,1\}^\ell \to \{0,1\}$ and parameters $m, \ell, d/\ell \in \mathbb{N}$, define the* Nisan-Wigderson generator $\mathrm{NW}^f_{m,d} \colon \{0,1\}^d \to \{0,1\}^m$ *as $\mathrm{NW}^f_{m,d}(z) := f(z_{S_1}) \cdots f(z_{S_m})$ for every $z \in \{0,1\}^d$, where $(S_1, \dots, S_m) := \mathcal{S}_{m,\ell,d}$.*

Nisan and Wigderson [43] showed that if f is a hard function (i.e. f cannot be approximated by small circuits) then $\mathrm{NW}^f_{m,d}$ is a pseudorandom generator secure against small circuits. The security proof of the Nisan-Wigderson generator transforms any statistical test for $\mathrm{NW}^f_{m,d}$ into a small circuit that approximately describes f. Moreover, as observed in [35], such small circuits can be constructed efficiently. We now make use of these facts to obtain a short description for f. Our proof is similar to the construction of Trevisan's extractor [50], but we need to argue the efficiency.

Lemma 20. *There exist some polynomial* poly *and a randomized polynomial-time oracle machine satisfying the following specification.*

Inputs: *A function $f \colon \{0,1\}^\ell \to \{0,1\}$ represented as its truth table, parameters $m, d/\ell, \delta^{-1} \in \mathbb{N}$ represented in unary, and oracle access to $T \subseteq \{0,1\}^m$.*

Promise: *We assume that the oracle T is a statistical test for $\mathrm{NW}^f_{m,d}$ with advantage δ. That is,*

$$\left| \Pr_{z \in_R \{0,1\}^d}\left[T(\mathrm{NW}^f_{m,d}(z)) = 1\right] - \Pr_{w \in_R \{0,1\}^m}\left[T(w) = 1\right] \right| \ge \delta. \tag{1}$$

Output: *A certificate for $\mathrm{K}^T_{t,\,\delta/2m}(f) \preceq \exp(\ell^2/d) \cdot m + d + O(\log(md))$, for any $t \ge \mathsf{poly}(m, d, 2^\ell)$.*

Proof. We first prove $\mathrm{K}^T_{t,\,\delta/m}(f) \le \exp(\ell^2/d) \cdot m + d + O(\log(md))$. We will then explain how to obtain a certificate efficiently (with the small loss in the quality δ/m of the approximation).

The first part is proved by a standard hybrid argument as in [43]. Without loss of generality, we may ignore the absolute value of (1); more precisely, let $T_b(w) := T(w) \oplus b$ for some $b \in \{0,1\}$ so that $\mathbb{E}_{z,w}\left[T_b(\mathrm{NW}^f_{m,d}(z)) - T_b(w)\right] \ge \delta$. For every $i \in [m]$, define a hybrid distribution $H_i := f(z_{S_1}) \cdots f(z_{S_i}) \cdot w_{i+1} \cdots w_m$ for $z \in_R \{0,1\}^d$ and $w \in_R \{0,1\}^m$. As H_0 and H_m are distributed identically to $w \in_R \{0,1\}^m$ and $\mathrm{NW}^f_{m,d}(z)$ for $z \in_R \{0,1\}^d$, respectively, we have $\mathbb{E}\left[T_b(H_m) - T_b(H_0)\right] \ge \delta$. Pick $i \in_R [m]$ uniformly at random. Then we obtain $\mathbb{E}_i\left[T_b(H_i) - T_b(H_{i-1})\right] \ge \delta/m$.

We can exploit this advantage to predict the next bit of the PRG (due to Yao [54]; a nice exposition can be found in [52, Proposition 7.16]). For each fixed $i \in [m]$, $c \in \{0,1\}$, $w_{[m]\setminus[i]} \in \{0,1\}^{m-i}$, and $z_{[d]\setminus S_i} \in \{0,1\}^{d-\ell}$, consider the following circuit P^{T_b} for predicting f: On input $x \in \{0,1\}^\ell$, set $z_{S_i} := x$ and construct $z \in \{0,1\}^d$. Output $T_b(f(z_{S_1}) \cdots f(z_{S_{i-1}}) \cdot c \cdot w_{i+1} \cdots w_m) \oplus c \oplus 1$. A basic idea here is that if $c = f(z_{S_i})\ \ (= f(x)\,)$ then the input distribution of T_b is identical to H_i and thus T_b is likely to output 1, in which case we should output c for predicting f. By a simple calculation, it can be shown that $\Pr[P^{T_b}(x) = f(x)] \ge \frac{1}{2} + \frac{\delta}{m}$, where the probability is taken over all $i \in_R [m]$, $c \in_R \{0,1\}$, $w_{[m]\setminus[i]} \in_R \{0,1\}^{m-i}$, $z_{[d]\setminus S_i} \in_R \{0,1\}^{d-\ell}$, and $x \in_R \{0,1\}^\ell$. In particular, by averaging, there exists some $i, c, w_{[m]\setminus[i]}, z_{[d]\setminus S_i}$ such that $\Pr_{x \in_R \{0,1\}^\ell}\left[P^{T_b}(x) = f(x)\right] \ge \frac{1}{2} + \frac{\delta}{m}$.

Therefore, it is sufficient to claim that the circuit P has a small description. Note that the value of f needed in the computation of P can be hardwired into the circuit using $\sum_{j<i} 2^{|S_i \cap S_j|}$ bits. Given oracle access to T, we can describe the $(\frac{1}{2} + \frac{\delta}{m})$-fraction of the truth table of f by specifying $m, \ell, d, b, c, i, w_{[m]\setminus[i]}, z_{[d]\setminus S_i}$, and the hardwired table of the values of f. This procedure takes time roughly $\mathsf{poly}(m,d) + \mathsf{poly}(2^\ell)$ (for computing the design and evaluating the entire truth table of P^{T_b}). The length of the description is at most $\sum_{j<i} 2^{|S_i \cap S_j|} + (m-i) + (d-\ell) + O(\log(md)) \le \exp(\ell^2/d) \cdot m + d + O(\log(md))$. Thus we have $\mathrm{K}^T_{t,\,\delta/m}(f) \le \exp(\ell^2/d) \cdot m + d + O(\log(md))$.

To find a certificate efficiently, observe that a random choice of $(c, i, w_{[m]\setminus[i]}, z_{[d]\setminus S_i})$ is sufficient in order for the argument above to work. That is, pick $c \in_R \{0,1\}$, $i \in_R [m]$, $w_{[m]\setminus[i]} \in_R \{0,1\}^{m-i}$, and $z_{[d]\setminus S_i} \in_R \{0,1\}^{d-\ell}$. Then a Markov style argument shows that, with probability at least $\delta/2m$, we obtain $\Pr_{x \in_R \{0,1\}^\ell}\left[P^{T_b}(x) = f(x)\right] \ge \frac{1}{2} + \frac{\delta}{2m}$. By trying each $b \in \{0,1\}$ and trying the random choice $O(m/\delta)$ times, we can find at least one certificate for $\mathrm{K}^T_{t,\delta/2m}(f)$ with high probability. ☐

We will update Lemma 20 by incorporating a list-decodable error-correcting code, so that we obtain a certificate for $\mathrm{K}^T_t(x)$ instead of $\mathrm{K}^T_{t,\delta/2m}(f)$.

Definition 21 (List-decodable error-correcting code; cf. [52]). *For every $n, m, L \in \mathbb{N}$ and $\epsilon > 0$, a function* $\mathrm{Enc}\colon \{0,1\}^n \to \{0,1\}^m$ *is called a* $(L, \frac{1}{2} - \epsilon)$*-list-decodable error-correcting code if there exists a function* $\mathrm{Dec}\colon \{0,1\}^m \to (\{0,1\}^n)^L$ *such that, for every $x \in \{0,1\}^n$ and $r \in \{0,1\}^m$ with* $\mathrm{dist}(\mathrm{Enc}(x), r) \le \frac{1}{2} - \epsilon$*, we have* $x \in \mathrm{Dec}(r)$*. We call* Dec *a* list decoder *of* Enc.

For our purpose, it is sufficient to use any standard list-decodable code such as the concatenation of a Reed-Solomon code and an Hadamard code.

Theorem 22 (see, e.g., [48] and [52, Problem 5.2]). *For any $n \in \mathbb{N}$ and $\epsilon > 0$, there exists a function* $\mathrm{Enc}_{n,\epsilon}\colon \{0,1\}^n \to \{0,1\}^{2^\ell}$ *with $\ell = O(\log(n/\epsilon))$ that is a $(\mathsf{poly}(1/\epsilon), \frac{1}{2} - \epsilon)$-list-decodable error-correcting code. Moreover,* $\mathrm{Enc}_{n,\epsilon}$ *and its list decoder* $\mathrm{Dec}_{n,\epsilon}$ *are computable in time* $\mathsf{poly}(n, 1/\epsilon)$*.*

In what follows, we implicitly regard a string $\mathrm{Enc}_{n,\epsilon}(x) \in \{0,1\}^{2^\ell}$ of length 2^ℓ as a function on ℓ-bit inputs.

Corollary 23. *$\mathrm{K}^A_{t'}(x) \le \mathrm{K}^A_{t,\epsilon}(\mathrm{Enc}_{n,\epsilon}(x)) + O(\log(n/\epsilon))$ for any string $x \in \{0,1\}^*$, any oracle A, and any $t' \ge t + \mathsf{poly}(n, 1/\epsilon)$. Moreover, given any x and any certificate for $\mathrm{K}^A_{t,\epsilon}(\mathrm{Enc}_{n,\epsilon}(x)) \preceq s$, one can find a certificate for $\mathrm{K}^A_{t'}(x) \preceq s + O(\log(n/\epsilon))$ in time $t + \mathsf{poly}(n, 1/\epsilon)$ with oracle access to A.*

Proof. Consider the following procedure M: Given input $d_0 \in \{0,1\}^*$ and $n, \epsilon^{-1} \in \mathbb{N}$ and index $i \in \mathbb{N}$, output the ith string of $\mathrm{Dec}_{n,\epsilon}(U^A(d_0))$. By the definition, there exists some description d_0 of length $\mathrm{K}^A_{t,\epsilon}(\mathrm{Enc}_{n,\epsilon}(x))$ such that $U^A(d_0)$ outputs some string r within time t and $\mathrm{dist}(\mathrm{Enc}_{n,\epsilon}(x), r) \le \frac{1}{2} - \epsilon$. Thus there exists an index $i \le \mathsf{poly}(1/\epsilon)$ such that the ith string of $\mathrm{Dec}_{n,\epsilon}(r)$ is equal to x. Hence, we obtain $\mathrm{K}^A_{t'}(x) \le |d_0| + O(\log(ni/\epsilon))$.

The "moreover" part can be easily seen as follows. Given a target string x and a description d_0, compute $\mathrm{Dec}_{n,\epsilon}(U^A(d_0))$. Let i be the index of x in the list $\mathrm{Dec}_{n,\epsilon}(U^A(d_0))$. Output a description $(M, d_0, n, \epsilon^{-1}, i)$. ☐

Now we combine Lemma 20 and the list-decodable error-correcting code.

Lemma 24. *There exist some polynomial poly and a randomized polynomial-time oracle machine satisfying the following specification.*

Inputs: *A string $x \in \{0,1\}^*$ of length $n \in \mathbb{N}$, parameters $m, d/\ell, \delta^{-1} \in \mathbb{N}$ represented in unary, and oracle access to $T \subseteq \{0,1\}^m$.*

Promise: *Let $\epsilon := \delta/2m$, and $2^\ell := |\mathrm{Enc}_{n,\epsilon}(x)|$. We assume that T is a statistical test for $\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}$ with advantage δ. That is,*

$$\left| \Pr_{z \in_R \{0,1\}^d} \left[T(\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}(z)) = 1 \right] - \Pr_{w \in_R \{0,1\}^m} \left[T(w) = 1 \right] \right| \ge \delta.$$

Output: *A certificate for $\mathrm{K}^T_t(x) \preceq \exp(\ell^2/d) \cdot m + d + O(\log(nmd/\delta))$ for any $t \ge \mathsf{poly}(n, m, d, 1/\delta)$.*

Proof. Set $f := \mathrm{Enc}_{n,\epsilon}(x) \in \{0,1\}^{2^\ell}$. Then run the algorithm of Lemma 20 with inputs $f, m, d/\ell$, δ^{-1} and oracle access to T. The algorithm outputs a certificate for $\mathrm{K}^T_{t_0,\,\delta/2m}(\mathrm{Enc}_{n,\epsilon}(x)) \preceq \exp(\ell^2/d) \cdot m + d + O(\log(md))$, where $t_0 = \mathsf{poly}(m, d, 2^\ell)$. By Corollary 23, we may efficiently convert this certificate to a certificate for $\mathrm{K}^T_t(x) \preceq \exp(\ell^2/d) \cdot m + d + O(\log(nmd/\epsilon))$, where $t \ge t_0 + \mathsf{poly}(n, 1/\epsilon)$. ☐

As a consequence of Lemma 24, for any $x \in \{0,1\}^*$ and parameters with $d \gg \ell^2$, we may obtain a certificate of length $\approx \exp(\ell^2/d) \cdot m + d \approx m + \ell^2 m/d + d$ given a statistical test for $\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}$. Setting $d := \ell\sqrt{m}$, we obtain a certificate of length $\approx m + O(\ell\sqrt{m})$. We now claim that m may be set to $\approx \mathrm{K}_t(x)$, by showing that the output of the Nisan-Wigderson generator is not random in the sense of time-bounded Kolmogorov complexity.

Lemma 25. *There exists some polynomial poly satisfying the following: For any $n, \epsilon^{-1}, m, d/\ell \in \mathbb{N}$, $z \in \{0,1\}^d$ and $x \in \{0,1\}^n$ (where 2^ℓ is the output length of $\mathrm{Enc}_{n,\epsilon}$), we have*

$$\mathrm{K}_{t'}(\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}(z)) \le \mathrm{K}_t(x) + d + O(\log(nmd/\epsilon))$$

for any $t, t' \in \mathbb{N}$ with $t' \geq t + \mathsf{poly}(n, 1/\epsilon, m, d)$.

Proof. Roughly speaking, the output of the Nisan-Wigderson generator can be described by a description d_0 of x (which is of length $\mathrm{K}_t(x)$), and a seed z of length d. More precisely, the following algorithm describes the output of the NW generator: Inputs consist of parameters $n, \epsilon^{-1}, m, d/\ell \in \mathbb{N}$ represented in binary, a seed $z \in \{0,1\}^d$, and a string $d_0 \in \{0,1\}^*$. The algorithm operates as follows. Compute $x := U(d_0)$, $f := \mathrm{Enc}_{n,\epsilon}(x)$, and the design $\mathcal{S}_{m,\ell,d}$. Output $\mathrm{NW}^f_{m,d}(z)$.

It is easy to see that the running time of this algorithm is at most $t + \mathsf{poly}(n, 1/\epsilon, m, d) \leq t'$, where t denotes the time it takes for $U(d_0)$ to output x. The length of the description is at most $|d_0| + |z| + O(\log(nmd/\epsilon)) \leq \mathrm{K}_t(x) + d + O(\log(nmd/\epsilon))$. □

We now assume that an oracle T is a δ-dense subset of r-random strings $R_r[t]$. By Lemma 25, T is a distinguisher for $\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}$ if $\mathrm{K}_t(x) + d \lesssim r(m)$. Thus by Lemma 24 we may find a certificate for $\mathrm{K}^T_{t'}(x) \precsim \exp(\ell^2/d) \cdot r^{-1}(\mathrm{K}_t(x) + d) + d$. A formal statement follows.

Theorem 26. *Let $r \colon \mathbb{N} \to \mathbb{N}$ be any function. There exist some polynomial poly and a randomized polynomial-time oracle machine satisfying the following specification.*

Inputs: *A string $x \in \{0,1\}^*$ of length $n \in \mathbb{N}$, parameters $t, m, d/\ell, \delta^{-1} \in \mathbb{N}$ represented in unary, and oracle access to $T \subseteq \{0,1\}^m$.*

Promise: *Let $\epsilon := \delta/2m$, and $2^\ell := |\mathrm{Enc}_{n,\epsilon}(x)|$. Assume that T is a δ-dense subset of $R_r[t_1]$ for some $t_1 \geq t + \mathsf{poly}(n, m, d, 1/\delta)$, and that $\mathrm{K}_t(x) + d + O(\log(nmd/\delta)) < r(m)$.*

Output: *A certificate for $\mathrm{K}^T_{t_2}(x) \preceq \exp(\ell^2/d) \cdot m + d + O(\log(nmd/\delta))$ for any $t_2 \geq \mathsf{poly}(n, m, d, 1/\delta)$.*

Proof. By Lemma 25, we have

$$\mathrm{K}_{t_1}(\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}(z)) \leq \mathrm{K}_t(x) + d + O(\log(nmd/\delta)) < r(m)$$

for any $z \in \{0,1\}^d$ and any $t_1 \geq t + \mathsf{poly}(n, m, d, 1/\delta)$. Therefore, for any $z \in \{0,1\}^d$, the output $\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}(z)$ is not r-random with respect to K_{t_1}; in particular, $\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}(z) \notin T$. On the other hand, $\Pr_{w \in_R \{0,1\}^m}[w \in T] \geq \delta$ by the assumption. Thus T distinguishes $\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}$ from the uniform distribution with advantage at least δ. By running the algorithm of Lemma 24 with input $x, m, d/\ell, \delta^{-1}$, and oracle access to T, the algorithm outputs a certificate for

$$\mathrm{K}^T_{t_2}(x) \preceq \exp(\ell^2/d) \cdot m + d + O(\log(nmd/\delta))$$

with high probability, where $t_2 \geq \mathsf{poly}(n, m, d, 1/\delta)$. □

By Theorem 26, for $r(m) \approx m$, we can set $m \approx \mathrm{K}_t(x) + d$; thus, we can find a certificate of length $\approx \exp(\ell^2/d) \cdot (\mathrm{K}_t(x) + d) + d \approx \mathrm{K}_t(x) + \ell^2 \mathrm{K}_t(x)/d + 2d + \ell^2$. By setting $d := \ell\sqrt{\mathrm{K}_t(x)}$, we obtain a certificate of length $\approx \mathrm{K}_t(x) + O(\ell\sqrt{\mathrm{K}_t(x)}) + \ell^2$. (Note here that we do not know a priori the best choice of d as well as $\mathrm{K}_t(x)$; however we can try all choices of d.) In the next corollary, we observe that the same length can be achieved as long as $m - O(\sqrt{m}\log m) \leq r(m)$.

Corollary 27. *Let $\delta^{-1} \in \mathbb{N}$ be any constant. Let $r \colon \mathbb{N} \to \mathbb{N}$ be any function such that $m - c\sqrt{m}\log m \leq r(m)$, for some constant c, for all large $m \in \mathbb{N}$. There exist some polynomial poly and a randomized polynomial-time oracle machine satisfying the following specification.*

Inputs: *A string $x \in \{0,1\}^*$ of length $n \in \mathbb{N}$, a parameter $t \in \mathbb{N}$ represented in unary, and oracle access to $T \subseteq \{0,1\}^*$.*

Promise: *For all large $m \in \mathbb{N}$, we assume that $T^{=m}$ is a δ-dense subset of $R_r[t_1]$ for some $t_1 \geq t + \mathsf{poly}(n)$.*

Output: *A certificate for $\mathrm{K}^T_{t_2}(x) \preceq \mathrm{K}_t(x) + O\big((\log n)\sqrt{\mathrm{K}_t(x)} + (\log n)^2\big)$ for any $t_2 \geq \mathsf{poly}(n)$.*

Proof. Without loss of generality, we may assume that $O(\log n) \leq \mathrm{K}_t(x) \leq n + O(1)$. Indeed, we may exhaustively search all the descriptions d_0 of length $O(\log n)$ and check if $U(d_0)$ outputs x within time t; if such a description is found, we may output d_0 as a certificate for $\mathrm{K}_t(x) \preceq |d_0| = O(\log n)$. Moreover, when $\mathrm{K}_t(x) \geq n + O(1)$, then we may just output a trivial description for x of length $n + O(1)$. In what follows, we assume $O(\log n) \leq \mathrm{K}_t(x) \leq n + O(1)$.

Here is the algorithm: For every $m \in \{1, \ldots, n + O(1)\}$ and every $d/\ell \in \{1, \ldots, n + O(1)\}$, run the algorithm of Theorem 26 on input $x, t, m, d/\ell, \delta^{-1}$ and with oracle access to $T^{=m}$. Output the shortest description found in this way. (If none is found, output a trivial description for x of length $n + O(1)$.)

We claim that, on some specific choice of $(m, d/\ell)$, the algorithm of Theorem 26 outputs a short description. Let $\ell := \log|\mathrm{Enc}_{n,\delta/2m}(x)| = O(\log n)$. We analyze the two cases depending on whether $\ell \leq \sqrt{\mathrm{K}_t(x)}$ or not. Consider the case when $\ell \leq \sqrt{\mathrm{K}_t(x)}$. Let $d/\ell := \sqrt{\mathrm{K}_t(x)}$ and $m := \mathrm{K}_t(x) + d + O(\log n) + 4c\sqrt{\mathrm{K}_t(x)}\log \mathrm{K}_t(x)$. Then we have $d \leq \mathrm{K}_t(x)$ and hence $m \leq 4\mathrm{K}_t(x)$. Thus,

$$\begin{aligned} r(m) &\geq m - c\sqrt{m}\log m \\ &\geq m - 2c\sqrt{\mathrm{K}_t(x)}\log 4\mathrm{K}_t(x) \\ &> \mathrm{K}_t(x) + d + O(\log n), \end{aligned}$$

which means the hypothesis of Theorem 26 is satisfied; therefore, with high probability, the algorithm outputs a description d_0 for x such that $|d_0| \leq \exp(\ell^2/d) \cdot m + d + O(\log n)$ with high probability. Since $\ell^2/d \leq 1$, the length of the description is

$$\begin{aligned} |d_0| &\leq (1 + 2\ell^2/d) \cdot m + d + O(\log n) \\ &\leq (1 + 2\ell^2/d) \cdot (\mathrm{K}_t(x) + d) + 3 \cdot (O(\log n) + 4c\sqrt{\mathrm{K}_t(x)}\log \mathrm{K}_t(x)) + d + O(\log n) \\ &\leq \mathrm{K}_t(x) + O(\ell\sqrt{\mathrm{K}_t(x)} + \ell^2). \end{aligned}$$

Next, consider the case when $\ell > \sqrt{\mathrm{K}_t(x)}$. In this case, let $d/\ell := \ell$ and $m := 4d$. Then we have $r(m) \geq m - c\sqrt{m}\log m \geq 3d > \mathrm{K}_t(x) + d + O(\log n)$, which confirms the hypothesis of Theorem 26. Thus the algorithm outputs a description for x of length $\exp(1) \cdot m + d + O(\log n) = O(\ell^2)$. □

4 In a world of Heuristica

In this section, we justify the hypothesis used in the previous section. We show that a dense r-random string can be accepted by some polynomial-time machine if $\mathsf{DistNP} \subseteq \mathsf{AvgP}$. To this end, we will define some specific problem $(\mathrm{MINKT}[r], \mathcal{D}^u)$ in DistNP.

4.1 Definitions

We first review background on average-case complexity. We consider the following distribution on a pair $(x, 1^t)$ of a binary string and a unary string.

Definition 28 (Uniform distribution with auxiliary unary input)**.** *Define a family of distributions $\mathcal{D}^u := \{\mathcal{D}^u_m\}_{m\in\mathbb{N}}$, where, for each $m \in \mathbb{N}$, $\mathcal{D}^u_m$ is defined as the output distribution of the following algorithm: Pick $n \in_R [m]$ and $x \in_R \{0,1\}^n$ randomly. Output $(x, 1^{m-n})$.*

By this definition, it is obvious that $\mathcal{D}^u$ is *efficiently samplable*: That is, there exists a randomized polynomial-time machine that, on input $1^m \in \mathbb{N}$, outputs a string according to the distribution $\mathcal{D}^u_m$.

Definition 29 (Distributional NP [31])**.** *For a language L and a family $\mathcal{D}$ of distributions, we say that $(L, \mathcal{D}) \in \mathsf{DistNP}$ if $L \in \mathsf{NP}$ and $\mathcal{D}$ is efficiently samplable.*

We consider a problem of deciding whether x is r-nonrandom string with respect to K_t on input $(x, 1^t)$ sampled from $\mathcal{D}^u$.

Definition 30. *For a function $r\colon \mathbb{N} \to \mathbb{N}$, define*

$$\mathrm{MINKT}[r] := \{\, (x, 1^t) \mid \mathrm{K}_t(x) < r(|x|) \,\}.$$

Fact 31. $(\mathrm{MINKT}[r], \mathcal{D}^u) \in \mathsf{DistNP}$ *if $r\colon \mathbb{N} \to \mathbb{N}$ is efficiently computable.*

We next define the notion of efficient algorithm solving a distributional problem.

Definition 32 (Errorless Heuristic Algorithm; cf. [14])**.** *Let $L \subseteq \{0,1\}^*$ be a language, $\mathcal{D} = \{\mathcal{D}_m\}_{m\in\mathbb{N}}$ be a family of distributions, and $\delta\colon \mathbb{N} \to [0,1]$. We say that an algorithm A is an* errorless heuristic algorithm for $(L, \mathcal{D})$ with failure probability δ *if*

- *$A(x)$ outputs either $L(x)$ or the special failure symbol $\perp$ for every $x \in \{0,1\}^*$, and*
- *$\Pr_{x\sim\mathcal{D}_m}[A(x) = \perp] \le \delta(m)$ for every m.*

We say that $(L, \mathcal{D}) \in \mathsf{Avg}_\delta\mathsf{P}$ if there exists an errorless heuristic deterministic polynomial-time algorithm for $(L, \mathcal{D})$ with failure probability δ. Define $\mathsf{AvgP} := \bigcap_{c\in\mathbb{N}} \mathsf{Avg}_{m^{-c}}\mathsf{P}$.

4.2 Errorless Algorithms and a Dense Subset of Random Strings

We claim that if $(\mathrm{MINKT}[r], \mathcal{D}^u)$ is easy on average then a dense subset of r-random strings can be accepted. For any oracle $T \subseteq \{0,1\}^*$ and any $t \in \mathbb{N}$, let T_t denote $\{\, x \in \{0,1\}^* \mid (x, 1^t) \in T \,\}$. The main idea here is that since there are few r-nonrandom strings, an errorless heuristic algorithm must succeed on a dense subset of r-random strings.

Lemma 33. *Let $r\colon \mathbb{N} \to \mathbb{N}$ be any function such that $r(n) < n$ for all large $n \in \mathbb{N}$. If $(\mathrm{MINKT}[r], \mathcal{D}^u) \in \mathsf{Avg}_\delta\mathsf{P}$ for $\delta(m) := 1/6m$, then there exists a language $T \in \mathsf{P}$ such that $T_t^{=n}$ is a $\frac{1}{3}$-dense subset of $R_t[r]$, for all large $n \in \mathbb{N}$ and every $t \in \mathbb{N}$.*

Proof. Let M be the errorless heuristic deterministic polynomial-time algorithm for $(\mathrm{MINKT}[r], \mathcal{D}^u)$. We define T so that $T(x, 1^t) := 1$ if $M(x, 1^t) = 0$; otherwise $T(x, 1^t) := 0$, for every $x \in \{0,1\}^*$ and $t \in \mathbb{N}$. By this definition, it is obvious that $T \in \mathsf{P}$.

Fix any $t \in \mathbb{N}$. We claim that T_t is a subset of r-random strings $R_t[r]$. Indeed, for any $x \in T_t$, we have $M(x, 1^t) = 0$. Since M is an errorless heuristic algorithm, we obtain $\mathrm{K}_t(x) \geq r(|x|)$; thus $x \in R_t[r]$.

We now claim that $T_t^{=n}$ is dense, i.e., $\Pr_{x \in_R \{0,1\}^n}[x \in T_t] \geq \frac{1}{3}$ for all large $n \in \mathbb{N}$. In the next claim, we prove that M solves $\mathrm{MINKT}[r]$ on average even if t is fixed.

Claim 34. *For all large $n \in \mathbb{N}$ and any $t \in \mathbb{N}$, we have $\Pr_{x \in_R \{0,1\}^n}\left[M(x, 1^t) \neq \mathrm{MINKT}[r](x, 1^t)\right] \leq \frac{1}{6}$.*

Indeed, for $m := n + t$, using the definition of errorless heuristic algorithms, we obtain

$$\begin{aligned}
\delta(m) &\geq \Pr_{(x,1^s) \sim \mathcal{D}^u_m}\left[M(x, 1^s) \neq \mathrm{MINKT}[r](x, 1^s)\right] \\
&\geq \Pr\left[|x| = n\right] \cdot \Pr\left[M(x, 1^s) \neq \mathrm{MINKT}[r](x, 1^s) \mid |x| = n\right] \\
&\geq \frac{1}{m} \cdot \Pr_{x \in_R \{0,1\}^n}\left[M(x, 1^t) \neq \mathrm{MINKT}[r](x, 1^t)\right],
\end{aligned}$$

where in the last inequality we used the fact that, conditioned on the event $|x| = n$, the distribution $\mathcal{D}^u_m$ is identically distributed to the distribution $(x, 1^t)$ where $x \in_R \{0,1\}^n$. Thus we have $\Pr_{x \in_R \{0,1\}^n}[M(x, 1^t) \neq \mathrm{MINKT}[r](x, 1^t)] \leq m \cdot \delta(m) \leq \frac{1}{6}$. This completes a proof of Claim 34.

We claim that M must output 0 on a large fraction of strings, which implies that T is dense. Indeed, there are few r-nonrandom strings, so M must succeed on a large fraction of random strings. More precisely, the number of r-nonrandom strings of length n is at most $\sum_{i=0}^{r(n)-1} 2^i \leq 2^{r(n)}$; thus, the probability that $(x, 1^t) \in \mathrm{MINKT}[r]$ over the choice of $x \in_R \{0,1\}^n$ is at most $2^{r(n)-n} \leq \frac{1}{2}$, for all large $n \in \mathbb{N}$ and every $t \in \mathbb{N}$. Therefore, we obtain

$$\begin{aligned}
\Pr_{x \in_R \{0,1\}^n}\left[x \in T_t\right] &= \Pr_x\left[M(x, 1^t) = 0 \ \&\ \mathrm{MINKT}[r](x, 1^t) = 0\right] \\
&= \Pr_x\left[M(x, 1^t) = \mathrm{MINKT}[r](x, 1^t)\right] - \Pr_x\left[M(x, 1^t) = 1 \ \&\ \mathrm{MINKT}[r](x, 1^t) = 1\right] \\
&\geq \left(1 - \frac{1}{6}\right) - \frac{1}{2} = \frac{1}{3}.
\end{aligned}$$

□

We will supply T_t to the algorithm of Corollary 27; then the algorithm will output some certificate under the oracle T_t. The next lemma enables us to convert the certificate to a certificate under the oracle T.

Lemma 35. *There exists some polynomial poly such that $\mathrm{K}^T_{t_3}(x) \leq \mathrm{K}^{T_{t_1}}_{t_2}(x) + O(\log\log t_1)$ for any $x \in \{0,1\}^*$, any oracle $T \subseteq \{0,1\}^*$ and any $t_1, t_2, t_3 \in \mathbb{N}$ such that t_1 is a power of 2 and $t_3 \geq \mathsf{poly}(t_1, t_2)$. Moreover, given $t_1 \in \mathbb{N}$ and a certificate for $\mathrm{K}^{T_{t_1}}_{t_2}(x) \preceq s$, one can efficiently find a certificate for $\mathrm{K}^T_{t_3}(x) \preceq s + O(\log\log t_1)$.*

Proof. Consider the following algorithm M with oracle access to T: Given input $d_0 \in \{0,1\}^*$ and $\log t_1 \in \mathbb{N}$, simulate and output $U^{T_{t_1}}(d_0)$. Here, the oracle T_{t_1} is simulated as follows: Given query q to T_{t_1}, convert it into a query $(q, 1^{t_1})$ to T.

If d_0 is a certificate for $\mathrm{K}_{t_2}^{T_{t_1}}(x) \preceq |d_0|$, then $M^T(d_0, \log t_1)$ outputs x in time $\mathsf{poly}(t_1, t_2)$. Thus $(M, d_0, \log t_1)$ is a certificate for $\mathrm{K}_{t_3}(x) \preceq |d_0| + O(\log\log t_1)$. $\square$

In order to obtain a *zero-error* randomized algorithm for GapMINKT, we prove that a Kolmogorov-random string can be used in order to derandomize randomized algorithms. This can be proved by using the Nisan-Wigderson generator or the Impagliazzo-Wigderson generator (cf. [43, 34, 37]); however, for our purpose, there is a much simpler construction of a pseudorandom generator based on Lemma 24. (Our construction is similar to the Sudan-Trevisan-Vadhan pseudorandom generator [48].)

Lemma 36. *For any constant $\gamma > 0$, there exist polynomials p_t, p_n and a constant $c \in \mathbb{N}$ satisfying the following: For all large $m \in \mathbb{N}$, let $t := p_t(m)$, $n := p_n(m)$, and $w \in \{0,1\}^n$ be a string such that $\mathrm{K}_t(w) \geq n^\gamma$. Then, $\mathrm{NW}_{m,d}^{\mathrm{Enc}_{n,\epsilon}(w)}$ is a pseudorandom generator secure against a circuit of size m, where $\epsilon := 1/4m$ and $d := c\log(nm)$; that is,*

$$\left| \Pr_{z \in_R \{0,1\}^d}\left[T(\mathrm{NW}_{m,d}^{\mathrm{Enc}_{n,\epsilon}(w)}(z)) = 1\right] - \Pr_{u \in_R \{0,1\}^m}\left[T(u) = 1\right] \right| < \frac{1}{2},$$

for any circuit T of size m.

Proof. Let $\ell := \log|\mathrm{Enc}_{n,\epsilon}(w)| = O(\log(nm))$. Let c be large enough so that $\exp(\ell^2/d) \leq (nm)^{\gamma/2}$ for all large $m \in \mathbb{N}$. By Lemma 24, if $\mathrm{NW}_{m,d}^{\mathrm{Enc}_{n,\epsilon}(w)}$ is not a pseudorandom generator secure against some circuit T of size m, then $\mathrm{K}_t^T(w) \leq \exp(\ell^2/d) \cdot m + d + O(\log(nm)) \leq (nm)^{\gamma/2} \cdot m + O(\log m)$ for $t := \mathsf{poly}(n, m, d)$. Since the circuit T can be described by a string of length $O(m \log m)$, we obtain $\mathrm{K}_t(w) \leq (nm)^{\gamma/2} \cdot m + O(m \log m)$. Thus we have $\mathrm{K}_t(w) < n^\gamma$ for some sufficiently large polynomials $n := p_n(m)$ and $t := p_t(m) \geq \mathsf{poly}(n, m, d)$, which is a contradiction. $\square$

We arrive at the following search to average-case reduction.

Restatement of Theorem 1. *Let $r\colon \mathbb{N} \to \mathbb{N}$ be any function such that for some constant $c > 0$, for all large $n \in \mathbb{N}$, $n - c\sqrt{n}\log n \leq r(n) < n$. Assume that $(\mathrm{MINKT}[r], \mathcal{D}^u) \in \mathsf{Avg}_{1/6m}\mathsf{P}$. Then, for some function $\sigma(n, s) = s + O\big((\log n)\sqrt{s} + (\log n)^2\big)$ and some polynomial τ, there exists a zero-error randomized polynomial-time algorithm solving the search version of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$.*

Proof. We first present a randomized algorithm that may err. By Lemma 33, there exists a language T in P such that $T_t^{=n}$ is a $\frac{1}{3}$-dense subset of $R_t[r]$ for all large $n \in \mathbb{N}$ and every $t \in \mathbb{N}$. Applying Corollary 27 to T_{t_1} and $\delta^{-1} = 3$, we obtain a randomized polynomial-time oracle machine that, on input x of length $n \in \mathbb{N}$, 1^t, and with oracle access to T_{t_1}, outputs a certificate d_0 for $\mathrm{K}_{t_2}^{T_{t_1}}(x) \preceq \sigma(n, \mathrm{K}_t(x))$ with high probability, for $t_1 \geq t + \mathsf{poly}(n)$ and $t_2 \geq \mathsf{poly}(n)$. On input $(x, 1^t)$, we fix t_1 to the minimum integer such that t_1 is a power of 2 and $t_1 \geq t + \mathsf{poly}(n)$. By Lemma 35, we can efficiently transform the certificate d_0 into a certificate d_1 for $\mathrm{K}_{t_3}^T(x) \preceq \sigma(n, \mathrm{K}_t(x)) + O(\log\log t_1)$, where $t_3 \geq \mathsf{poly}(t_1, t_2)$. Since T is solvable by a deterministic polynomial-time machine, by Lemma 14, we can efficiently transform the certificate d_1 into a certificate d_2 for $\mathrm{K}_{t_4}(x) \preceq \sigma(n, \mathrm{K}_t(x)) +$

$O(\log\log t_1)$, where $t_4 \geq \mathsf{poly}(t_3)$. Thus we obtain a randomized polynomial-time algorithm that, on input $(x, 1^t)$, outputs a certificate d_2 for $\mathrm{K}_{t_4}(x) \preceq \sigma(|x|, \mathrm{K}_t(x)) + O(\log\log t_1)$ where $t_1 = \Theta(t + \mathsf{poly}(|x|))$ and $t_4 \geq \tau(|x|, t)$ for some polynomial τ.

Note that there is an additive term $O(\log\log t_1)$; however, we may assume without loss of generality that $O(\log\log t_1) = O(\log n)$, which can be absorbed into $\sigma(n, \mathrm{K}_t(x))$. Indeed, otherwise we have $t_1 \geq 2^n$, and hence $t \geq \Omega(2^n)$. In such a case, we can exhaustively search all the description in time $\mathsf{poly}(t) = 2^{O(n)}$, and we can find the shortest description.

Now we make the randomized algorithm zero-error. Let M denote the randomized algorithm solving the search version of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$. Fix any input $(x, 1^t)$. Let k, m, t' be some large polynomials in $|x|$ and t that will be chosen later. Pick $w \in_R \{0,1\}^k$ uniformly at random, and check if $w \in T_{t'}^{=k}$; note that w is r-random since $T_{t'}^{=k}$ is a subset of $R_{t'}[r]$. Since $T_{t'}^{=n}$ is dense, we can find such an r-random string with high probability; if no r-random string is found, then output $\perp$ and halt (i.e., the zero-error algorithm fails). Using w as a source of a hard function, we construct the secure pseudorandom generator $\mathrm{NW}_{m,d}^{\mathrm{Enc}_{k,\epsilon}(w)}$ given in Lemma 36. Since the seed length of the pseudorandom generator is $O(\log(km))$, we can enumerate all the seeds z in polynomial time; we use $\mathrm{NW}_{m,d}^{\mathrm{Enc}_{k,\epsilon}(w)}(z) \in \{0,1\}^m$ as the source of randomness of the randomized algorithm M. Output the shortest description that is found by exhaustively searching all the seeds.

To prove the correctness, we define some statistical test T. Fix any input $(x, 1^t)$, and let C_M be a polynomial-size circuit that takes random bits u and simulates the randomized algorithm M on input $(x, 1^t)$ with random bits u. Let $s := \sigma(|x|, \mathrm{K}_t(x))$, $t'' := \tau(|x|, t)$, and let V be a polynomial-size circuit that takes a description d_0 and accepts iff d_0 is a certificate for $\mathrm{K}_{t''}(x) \preceq s$. Define a statistical test T as $T(u) := V(C_M(u))$. Let m be large enough so that $|T| \leq m$ (for every choice of s); define $k := p_n(m)$ and $t' := p_t(m)$ where p_n and p_t are polynomials in Lemma 36. Since M finds a certificate with high probability, we have $\Pr_{u \in_R \{0,1\}^m}[T(u) = 1] \geq \frac{1}{2}$. Thus by Lemma 36, there exists some seed $z \in \{0,1\}^d$ such that $C_M(\mathrm{NW}_{m,d}^{\mathrm{Enc}_{k,\epsilon}(w)}(z))$ outputs a certificate for $\mathrm{K}_{t''}(x) \preceq s$, as desired. ☐

Corollary 37. *In the following list, we have* $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 4$. *Moreover, if* Promise-ZPP = Promise-P*, then we also have* $4 \Rightarrow 2$.

1. $\mathsf{DistNP} \subseteq \mathsf{AvgP}$.
2. $(\mathrm{MINKT}[r], \mathcal{D}^u) \in \mathsf{Avg}_{1/6m}\mathsf{P}$ *for some* $r\colon \mathbb{N} \to \mathbb{N}$ *such that* $n - O\big(\sqrt{n}\log n\big) \leq r(n) < n$ *for all large* $n \in \mathbb{N}$.
3. *There exists a zero-error randomized polynomial-time algorithm solving the search version of* $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$*, for some* $\sigma(n,s) = s + O\big((\log n)\sqrt{s} + (\log n)^2\big)$ *and some polynomial* $\tau(n,t)$.
4. $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in$ Promise-ZPP *for some* $\sigma(n,s) = s + O\big((\log n)\sqrt{s} + (\log n)^2\big)$ *and some polynomial* $\tau(n,t)$.

Proof. ($1 \Rightarrow 2$) For $r(n) := n - 1$, we have $(\mathrm{MINKT}[r], \mathcal{D}^u) \in \mathsf{DistNP} \subseteq \mathsf{AvgP} \subseteq \mathsf{Avg}_{1/6m}\mathsf{P}$.

($2 \Rightarrow 3$) This is exactly equivalent to Theorem 1.

($3 \Rightarrow 4$) This follows from Fact 13.

($4 \Rightarrow 2$ if Promise-ZPP = Promise-P) Under the derandomization assumption, we have $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in$ Promise-ZPP = Promise-P. Let c be a constant such that $\sigma(n,s) \leq s +$

$c \cdot \left((\log n)\sqrt{s} + (\log n)^2\right)$ for all large $n, s \in \mathbb{N}$. We claim that $(\text{MINKT}[r], \mathcal{D}^u) \in \mathsf{Avg}_\delta\mathsf{P}$ for $r(n) := n - 2c(\log n)\sqrt{n}$ and $\delta(m) := 1/6m$.

By the assumption, there exists a deterministic polynomial-time algorithm M that distinguishes YES and NO instances of $\text{Gap}_{\sigma,\tau}\text{MINKT}$. Using the algorithm, we define an errorless heuristic algorithm A solving MINKT$[r]$ as follows: On input $(x, 1^t)$, if the input is short, i.e., $|x| = O(1)$, then check if $(x, 1^t) \in \text{MINKT}[r]$ by an exhaustive search, and output the answer. Otherwise, set $s := r(|x|)$ and output 0 if $M(x, 1^t, 1^s)$ rejects; otherwise, output $\perp$.

We claim that A is errorless. Since A does not output 1 on inputs of large length, it suffices to claim that $M(x, 1^t, 1^s)$ accepts for any $(x, 1^t) \in \text{MINKT}[r]$. Since $\mathrm{K}_t(x) < r(|x|) = s$, $(x, 1^t, 1^s)$ is an YES instance of $\text{Gap}_{\sigma,\tau}\text{MINKT}$; thus $M(x, 1^t, 1^s)$ accepts.

We claim that A succeeds on a large fraction of inputs. Fix any large $n \in \mathbb{N}$ and any $t \in \mathbb{N}$. For any $x \in \{0,1\}^n$, $A(x, 1^t)$ outputs $\perp$ only if $(x, 1^t, 1^{r(n)})$ is *not* a NO instance of $\text{Gap}_{\sigma,\tau}\text{MINKT}$. Hence,

$$\begin{aligned}\Pr_{x \in_R \{0,1\}^n}\left[A(x, 1^t) = \perp\right] &\le \Pr_{x \in_R \{0,1\}^n}\left[\mathrm{K}_{\tau(n,t)}(x) \le \sigma(n, r(n))\right] \\ &\le 2^{-n+\sigma(n,r(n))+1} \le \frac{1}{6},\end{aligned}$$

where the last inequality holds for all large n. Thus, for every $m \in \mathbb{N}$, we obtain

$$\Pr_{(x,1^t) \sim \mathcal{D}^u_m}\left[A(x, 1^t) = \perp\right] = \mathop{\mathbb{E}}_{n \in_R [m]}\left[\Pr_{x \in_R \{0,1\}^n}\left[A(x, 1^{m-n}) = \perp\right]\right] \le \frac{1}{6}.$$

□

5 Hardness of MINKT

In this section, we present evidence against $\text{Gap}_{\sigma,\tau}\text{MINKT} \in \mathsf{coNP}$. We start with the definition of hitting set generator, which is a stronger notion than pseudorandom generator.

Definition 38 (Hitting set generators). *Let $\gamma\colon \mathbb{N} \to [0,1]$ be a function. Let $G := \{G_n\colon \{0,1\}^{s(n)} \to \{0,1\}^{t(n)}\}_{n\in\mathbb{N}}$ be a family of functions. A promise problem $(L_{\mathrm{Y}}, L_{\mathrm{N}})$ is said to γ-avoid G if for every $n \in \mathbb{N}$, $G_n(z) \in L_{\mathrm{N}}$ for any $z \in \{0,1\}^{s(n)}$, and $\Pr_{w \in_R \{0,1\}^{t(n)}}\left[w \in L_{\mathrm{Y}}\right] \ge \gamma(n)$. G is called a* hitting set generator γ-secure *against a complexity class $\mathfrak{C}$ if there is no promise problem $(L_{\mathrm{Y}}, L_{\mathrm{N}}) \in \mathfrak{C}$ that γ-avoids G.*

For a hitting set generator, we measure the time complexity with respect to the output length $t(n)$; that is, we say that a family of functions $G := \{G_n\colon \{0,1\}^{s(n)} \to \{0,1\}^{t(n)}\}_{n\in\mathbb{N}}$ is *efficiently computable* if there exists a polynomial-time algorithm that, on input $z \in \{0,1\}^{s(n)}$, computes $G_n(z)$ in time $\mathsf{poly}(t(n))$ for all large $n \in \mathbb{N}$.

Note that there is no efficiently computable hitting set generator γ-secure against coNP for any "admissible" γ. On the other hand, as we will see, it is conjectured that there exists a hitting set generator secure against NP. We first claim that there is no hitting set generator secure against P^A for any oracle A solving GapMINKT. For simplicity, we focus on the case of $t(n) = n$.

Theorem 39. *Let $\sigma, \tau\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be any functions such that $\sigma(n, s) \ge s$ for any $n, s \in \mathbb{N}$. Let $G = \{G_n\colon \{0,1\}^{s(n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ be any family of functions computable in time $\mathsf{poly}(n)$,* *where $s\colon \mathbb{N} \to \mathbb{N}$ is an efficiently computable function. Let $\gamma\colon \mathbb{N} \to [0,1]$ be any function such that $\sigma(n, s(n) + O(\log n)) \le n - 1 + \log(1 - \gamma(n))$ for any $n \in \mathbb{N}$. Then, there exists a deterministic polynomial-time oracle machine M (in fact, a one-query reduction) such that M^A γ-avoids G for any oracle $A \subseteq \{0,1\}^*$ solving the promise problem* $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$.

Proof. Since $G_n(z)$ can be described by its seed $z \in \{0,1\}^{s(n)}$ and an integer $n \in \mathbb{N}$ in time $\mathsf{poly}(n)$, we have $\mathrm{K}_t(G_n(z)) \le s(n) + O(\log n)$ for every $n \in \mathbb{N}$, every $z \in \{0,1\}^{s(n)}$ and $t := \mathsf{poly}(n)$. Let $s'(n) := s(n) + O(\log n)$ denote the upper bound on $\mathrm{K}_t(G_n(z))$.

The algorithm of M is defined as follows: On input $x \in \{0,1\}^*$ of length n, accept iff $(x, 1^{\mathsf{poly}(n)}, 1^{s'(n)}) \notin A$.

We claim that $M^A(G_n(z)) = 0$ for any $n \in \mathbb{N}$ and any $z \in \{0,1\}^{s(n)}$. Since $(G_n(z), 1^{\mathsf{poly}(n)}, 1^{s'(n)})$ is an Yes instance of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$, we have $A(G_n(z), 1^{\mathsf{poly}(n)}, 1^{s'(n)}) = 1$; hence $M^A(G(z)) = 0$.

On the other hand, we claim that $M^A(w) = 1$ for every $n \in \mathbb{N}$ and for most $w \in \{0,1\}^n$. Note that $M^A(w) = 1$ if $(w, 1^{\mathsf{poly}(n)}, 1^{s'(n)})$ is a No instance of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$; that is, $\mathrm{K}_{t'}(w) > \sigma(n, s'(n))$ for $t' := \tau(n, \mathsf{poly}(n))$. The number of $w \in \{0,1\}^n$ such that $\mathrm{K}_{t'}(w) > \sigma(n, s'(n))$ is at least $2^n - 2^{\sigma(n,s'(n))+1}$. Thus, the probability that $M^A(w) = 1$ over the choice of $w \in_R \{0,1\}^n$ is at least $1 - 2^{\sigma(n,s'(n))+1-n} \ge \gamma(n)$. ☐

In particular, for the parameter $\sigma(n,s) := s + O\big((\log n)\sqrt{s} + (\log n)^2\big)$ of Theorem 1, $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ is capable of avoiding any efficiently computable hitting set generator $G = \{G_n\colon \{0,1\}^{s(n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ such that $s(n) \le n - c\sqrt{n}\log n$ for some large constant $c > 0$. In what follows, we present specific candidate hitting set generators conjectured to be secure against NP.

5.1 Natural Properties and Rudich's Conjecture

Natural properties, introduced by Razborov and Rudich [46], can be cast as algorithms breaking a particular hitting set generator. The hitting set generator is defined as follows.

Definition 40 (Circuit interpreter)**.** *Let $s\colon \mathbb{N} \to \mathbb{N}$ be a function. Let*

$$G^{\mathsf{int},s} := \{G^{\mathsf{int},s}_\ell\colon \{0,1\}^{O(s(\ell)\log s(\ell))} \to \{0,1\}^{2^\ell}\}_{\ell\in\mathbb{N}}$$

denote the family of circuit interpreters *$G^{\mathsf{int},s}_\ell$ with parameter s, defined as follows: $G^{\mathsf{int},s}_\ell$ takes as input a description $z_C \in \{0,1\}^{O(s(\ell)\log s(\ell))}$ of a circuit C of size at most $s(\ell)$ on ℓ inputs, and outputs the truth table of the function computed by C.*

Definition 41 (Γ-natural property)**.** *A promise problem $(L_{\mathrm{Y}}, L_{\mathrm{N}})$ is called a* natural property useful against $\mathsf{SIZE}(s(\ell))$ with largeness γ *if $(L_{\mathrm{Y}}, L_{\mathrm{N}})$ γ-avoids the circuit interpreter $G^{\mathsf{int},s}$ with parameter s. If, in addition, $(L_{\mathrm{Y}}, L_{\mathrm{N}}) \in$ Promise-Γ for a complexity class Γ such as P, BPP or NP, then $(L_{\mathrm{Y}}, L_{\mathrm{N}})$ is called a* Γ-natural property.

Rudich [47] conjectured that there is no $\mathsf{NP/poly}$-natural property useful against $\mathsf{P/poly}$. In our terminology, his conjecture implies that $G^{\mathsf{int},s}$ is a hitting set generator secure against $\mathsf{NP/poly}$ for any $s(\ell) = \ell^{\omega(1)}$. Thus his conjecture implies $\mathrm{Gap}_{\sigma,\tau}\mathrm{MKTP} \notin \mathsf{coNP/poly}$ for a wide range of parameters σ.

Corollary 42. *Let $s(n) = (\log n)^{\omega(1)}$ for $n \in \mathbb{N}$. Let $\sigma, \tau\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be any functions such that $\sigma(n, s(n) + O(\log n)) \le n - 2$ for any $n \in \mathbb{N}$. If $\mathrm{Gap}_{\sigma,\tau}\mathrm{MKTP} \in \mathsf{coNP/poly}$, then there is some $\mathsf{NP/poly}$-natural property useful against $\mathsf{P/poly}$ with largeness $\frac{1}{2}$.*

Proof. We apply Theorem 39 to the circuit interpreter $G^{\mathsf{int},s'}$ for $s'(\ell) := s(2^\ell)$. Then we obtain an $\mathsf{NP}/\mathsf{poly}$ algorithm A that $\frac{1}{2}$-avoids $G^{\mathsf{int},s'}$. We claim that A computes a natural property useful against P/poly in the original sense of Razborov and Rudich [46]: The $\mathsf{NP}/\mathsf{poly}$ *constructivity* is obvious because A is an $\mathsf{NP}/\mathsf{poly}$ algorithm. We also have *largeness* $\frac{1}{2}$ by the definition. It remains to claim the *usefulness* against P/poly: Since $s'(\ell)$ is a super-polynomial in $\ell \in \mathbb{N}$, the image of $G_\ell^{\mathsf{int},s'}$ is a superset of truth tables of functions computable by P/poly. Hence A does not accept any truth table computable by P/poly (for all large input length), which means that A is useful against P/poly. $\square$

5.2 Random 3SAT-Hardness

More significantly, we can also prove that $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in \mathsf{coNP}$ implies that Random 3SAT is easy for a coNP algorithm. This is due to the fact that Random 3SAT can be seen as another particular hitting set generator $G = \{G_n : \{0,1\}^{n-\Omega(n/\log n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$.

We define a random 3SAT problem as a distributional NP problem. Let Δ be a sufficiently large constant $(> 1/\log(8/7) \approx 5.19)$. For the number n of variables, let $m := \Delta n$ be the number of clauses. The distribution is defined as follows. For each $i \in [m]$, choose a clause C_i randomly out of the $8n^3$ possible clauses of 3CNFs (for each choice of 3 variables with replacement, we have $2^3 = 8$ ways to negate the variables). Output a 3CNF formula $\varphi := \bigwedge_{i=1}^m C_i$. We regard $\varphi \in \{0,1\}^N$ for $N := m\log(8n^3)$.[3] Let $\mathcal{D}_{\mathsf{3SAT}}$ denote the distribution defined in this way (which is in fact the uniform distribution on $\{0,1\}^N$ if $N \in \mathbb{N}$). Then, Random 3SAT is defined as the distributional problem $(\mathsf{3SAT}, \mathcal{D}_{\mathsf{3SAT}})$.

Theorem 43. *Let $\sigma, \tau : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be any functions such that, for any constant $c_0 > 0$, for some constant $c_1 > 0$, for all large $N \in \mathbb{N}$, $\sigma(N, N - c_0 N/\log N) \le N - c_1 N/\log N$. Then, $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ is Random 3SAT-hard. In particular, if $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in \mathsf{coNP}$, then there exists an errorless heuristic coNP algorithm solving Random 3SAT with failure probability $\le 2^{-\Omega(n)}$, where n denotes the number of variables.*

Proof. For all large $N \in \mathbb{N}$, we claim that there exists an efficiently computable function $G_N : \{0,1\}^{s(N)} \to \{0,1\}^N$ such that the image of G_N contains all the satisfiable formulas of n variables and m clauses. Indeed, any satisfiable formula can be described by using an assignment $a \in \{0,1\}^n$, and an $m\log(7n^3)$-bit string for describing m clauses satisfied by a; that is, given an assignment $a \in \{0,1\}^n$ and 3 variables, there are 7 ways to negate these variables so that the resulting clause is satisfied by a. Thus we may set $s(N) = n + m\log(7n^3) = N - (\Delta\log(8/7) - 1)n = N - \Omega(n) \le N - \Omega(N/\log N)$.

Applying Theorem 39 to $G = \{G_N\}_{N\in\mathbb{N}}$, we obtain a polynomial-time algorithm M with oracle access to $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ that γ-avoids G for $\sigma(N, s(N) + O(\log N)) \le N - 1 + \log(1 - \gamma(N))$. Since $\sigma(N, s(N) + O(\log N)) \le N - \Omega(N/\log N) = N - \Omega(n)$, we have $\gamma(N) = 1 - 2^{-\Omega(n)}$; therefore, the algorithm M solves Random 3SAT with failure probability $\le 2^{-\Omega(n)}$. $\square$

[3] In order to cast Random 3SAT as a hitting set generator, we assume that n is a power of 2 so that $N \in \mathbb{N}$. This assumption is however not crucial, and it is easy to observe that Random 3SAT is easy on average for *every* $n \in \mathbb{N}$ with oracle access to $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ using the algorithm of Theorem 39.

6 Worst-Case to Average-Case Reduction for MCSP

In this section, we establish a worst-case and average-case equivalence for approximating a minimum circuit size. We start by introducing the problem.

Definition 44 (GapMCSP)**.** *For any constant $\epsilon \in (0, 1]$, the promise problem* $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ *is defined as follows: The input consists of a function $f\colon \{0,1\}^n \to \{0,1\}$ represented as its truth table (of length 2^n) and an integer $s \in \mathbb{N}$. The task is to distinguish the* YES *instances (f, s) such that $\mathsf{size}(f) \le s$, and the* NO *instances (f, s) such that $\mathsf{size}(f) > 2^{(1-\epsilon)n} \cdot s$.*

When $\epsilon = 1$, $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ corresponds to the Minimum Circuit Size Problem (MCSP). There is a natural search version associated to the promise problem.

Definition 45 (Search version of GapMCSP)**.** *The search version of* $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ *is defined as follows: On input a function $f\colon \{0,1\}^n \to \{0,1\}$ represented as its truth table, the task is to output a circuit C such that C computes f and $|C| \le 2^{(1-\epsilon)n} \cdot \mathsf{size}(f)$.*

Fact 46 (Decision reduces to search for MCSP)**.** *If there exists a randomized algorithm solving the search version of* $\mathrm{Gap}_\epsilon\mathrm{MCSP}$*, then* $\mathrm{Gap}_\epsilon\mathrm{MCSP} \in \text{Promise-}\mathsf{RP}$.

Proof Sketch. This is essentially the same with Fact 13. On input (f, s), run the search algorithm on input f to obtain some circuit C. Accept if and only if C computes f and $|C| \le 2^{(1-\epsilon)n} \cdot s$. □

We consider the distributional NP problem of the following problem under the uniform distribution.

Definition 47 (Parameterized Minimum Circuit Size Problem)**.** *For a function $s\colon \mathbb{N} \to \mathbb{N}$,* the Minimum Circuit Size Problem with parameter s*, abbreviated as* $\mathrm{MCSP}[s]$*, is the following problem: Given a function $f\colon \{0,1\}^n \to \{0,1\}$ represented as its truth table, decide whether $\mathsf{size}(f) \le s(n)$.*

Let $\mathcal{U} := \{U_n\}_{n\in\mathbb{N}}$ be the family of the uniform distributions U_n on $\{0,1\}^n$, for each $n \in \mathbb{N}$. We next define randomized errorless heuristic algorithms; for simplicity, we focus on the case of the uniform distribution.

Definition 48 (Randomized Errorless Heuristics; cf. [14])**.** *Let $(L, \mathcal{U})$ be a distributional problem and $\delta\colon \mathbb{N} \to [0, 1]$. A randomized algorithm A is said to be a* randomized errorless heuristic algorithm *with failure probability δ for $(L, \mathcal{U})$ if*

- $\Pr_A\left[A(x) \notin \{L(x), \bot\}\right] \le \frac{1}{8}$ *for every $x \in \{0,1\}^*$, and*
- $\Pr_{x\sim U_n}\left[\Pr_A[A(x) = \bot] > \frac{1}{8}\right] \le \delta(n)$ *for every $n \in \mathbb{N}$.*

An input x such that $\Pr_A[A(x) = \bot] > \frac{1}{8}$ is called a hard instance *for A. We say that $(L, \mathcal{U}) \in \mathsf{Avg}_\delta\mathsf{BPP}$ if $(L, \mathcal{U})$ admits a randomized polynomial-time errorless heuristic algorithm with failure probability δ. Define $\mathsf{AvgBPP} := \bigcap_{c\in\mathbb{N}} \mathsf{Avg}_{n^{-c}}\mathsf{BPP}$.*

Using the insight from [27], we show that an errorless heuristic algorithm for $\mathrm{MCSP}[s]$ is essentially equivalent to BPP-natural properties useful against $\mathsf{SIZE}(s(n))$.

Lemma 49. *Let $s\colon \mathbb{N} \to \mathbb{N}$ be any function such that $s(n) = o(2^n/n)$ for $n \in \mathbb{N}$. Let $\gamma, \delta\colon \mathbb{N} \to [0, 1]$ be functions.*

1. *If there exists a* BPP*-natural property useful against* $\mathsf{SIZE}(s(n))$ *with largeness* γ*, then* $(\mathrm{MCSP}[s],\mathcal{U}) \in \mathsf{Avg}_\delta\mathsf{BPP}$*, where* $\delta(2^n) := 1 - \gamma(n)$ *for* $n \in \mathbb{N}$*.*

2. *If* $(\mathrm{MCSP}[s],\mathcal{U}) \in \mathsf{Avg}_\delta\mathsf{BPP}$*, then there exists a* BPP*-natural property useful against* $\mathsf{SIZE}(s(n))$ *with largeness* γ *where* $\gamma(n) = 1 - \delta(2^n) - 2^{-2^{n-1}}$ *for* $n \in \mathbb{N}$*.*

Proof. First part: Let $(L_\mathrm{Y}, L_\mathrm{N})$ be a BPP-natural property, and M be a BPP algorithm solving $(L_\mathrm{Y}, L_\mathrm{N})$ (with error $\le \frac{1}{8}$). Define a randomized algorithm A as follows: On input f, run M on input f and reject if M accepts, and output $\perp$ otherwise. We claim that A is a randomized errorless heuristic algorithm for $(\mathrm{MCSP}[s],\mathcal{U})$.

We first claim that the fraction of hard instances for A is small. Indeed, f is a hard instance for A only if $\Pr_A[A(f) = \perp] = \Pr_M[M(f) = 0] > \frac{1}{8}$, which implies that $f \notin L_\mathrm{Y}$. Thus the fraction of hard instances $f \in \{0,1\}^{2^n}$ is at most $1 - \gamma(n)$.

Next, we claim that, for every input f, A outputs a wrong answer for $\mathrm{MCSP}[s]$ with probability at most $\frac{1}{8}$. Since A never accepts, this happens only if M accepts and $f \in \mathrm{MCSP}[s]$, which implies that $f \in L_\mathrm{N}$ and hence $\Pr_A[A(f) = 0] = \Pr_M[M(f) = 1] \le \frac{1}{8}$.

Second part: Given a randomized errorless heuristic algorithm A for $(\mathrm{MCSP}[s],\mathcal{U})$, define a randomized algorithm M so that $M(f) := 0$ if $A(f) = 1$ or $A(f) = \perp$; otherwise $M(f) := 1$. We claim that M accepts some natural property $(L_\mathrm{Y}, L_\mathrm{N})$ with error $\le \frac{1}{4}$.

Since A is errorless, for any $f \in \mathrm{MCSP}[s]$, we have $\Pr_M[M(f) = 1] = \Pr_A[A(f) = 0] \le \frac{1}{8}$; thus M satisfies the usefulness. To see the largeness, consider any instance f that is not a hard instance for A. We claim that $\Pr_A[A(f) = \mathrm{MCSP}[s](f)] \ge \frac{3}{4}$: This is because A outputs $\perp$ with probability at most $\frac{1}{8}$ since f is not hard for A, and moreover A outputs a wrong answer with probability at most $\frac{1}{8}$. Therefore, M accepts f with probability at least $\frac{3}{4}$ if f is not a hard instance for A and $f \notin \mathrm{MCSP}[s]$. The fraction of such instances $f \in \{0,1\}^{2^n}$ is at least $1 - \delta(2^n) - s(n)^{O(s(n))} \cdot 2^{-2^n} \ge \gamma(n)$, for all large $n \in \mathbb{N}$. □

We now state the main result of this section.

Theorem 50. *The following are equivalent.*

1. $\mathrm{Gap}_\epsilon\mathrm{MCSP} \in \text{Promise-}\mathsf{BPP}$ *for some* $\epsilon > 0$*.*

2. $(\mathrm{MCSP}[2^{\epsilon n}],\mathcal{U}) \in \mathsf{AvgBPP}$ *for some* $\epsilon > 0$*.*

3. $(\mathrm{MCSP}[2^{\epsilon n}],\mathcal{U}) \in \mathsf{Avg}_\delta\mathsf{BPP}$ *for some constants* $\epsilon, \delta \in (0,1)$*.*

4. *There exists a* BPP*-natural property useful against* $\mathsf{SIZE}(2^{\epsilon n})$ *with largeness* γ*, for some* $\epsilon \in (0,1)$ *and* $\gamma(n) := 1 - 2^{-2^{n-1}}$*.*

5. *There exists a* BPP*-natural property useful against* $\mathsf{SIZE}(2^{\epsilon n})$ *with largeness* γ*, for some constants* $\epsilon, \gamma \in (0,1)$*.*

6. *There exists a randomized polynomial-time algorithm solving the search version of* $\mathrm{Gap}_\epsilon\mathrm{MCSP}$*, for some* $\epsilon > 0$*.*

Proof. (5 $\Leftrightarrow$ 3 and 4 $\Rightarrow$ 2) This follows from Lemma 49.

(2 $\Rightarrow$ 3) Obvious.

(6 $\Rightarrow$ 1) This follows from Fact 46.

(1 $\Rightarrow$ 4) Let A be a randomized polynomial-time algorithm solving $\mathrm{Gap}_\epsilon\mathrm{MCSP}$. Define A' as the following algorithm: On input $f\colon \{0,1\}^n \to \{0,1\}$, run A on input (f,s) for $s := 2^{\epsilon n/2}$, and accept iff A rejects. We claim that A' accepts some natural property useful against $\mathsf{SIZE}(2^{\epsilon n/2})$. Indeed, if $\mathsf{size}(f) \le 2^{\epsilon n/2}$, then (f,s) is an YES instance of $\mathrm{Gap}_\epsilon\mathrm{MCSP}$, and thus $A'(f)$ rejects with high probability. Hence A' satisfies the usefulness. On the other hand, A' accepts any NO instance (f,s) of $\mathrm{Gap}_\epsilon\mathrm{MCSP}$, that is, any (f,s) such that $\mathsf{size}(f) > 2^{(1-\epsilon)n}\cdot s = 2^{(1-\epsilon/2)n}$. Since the fraction of functions f such that $\mathsf{size}(f) \le 2^{(1-\epsilon/2)n}$ is at most $2^{O(n2^{(1-\epsilon/2)n})-2^n} \le 2^{-2^n/2}$, A' satisfies the largeness of density $1-2^{-2^n/2}$.

(5 $\Rightarrow$ 6) This is the main technical part, which can be proved by using a generic reduction from learning to natural properties [17]. We prove this in the next Theorem 51. $\square$

Theorem 51. *If there exists a BPP-natural property useful against $\mathsf{SIZE}(2^{\epsilon_0 n})$ with largeness δ_0 for some constants $\epsilon_0, \delta_0 \in (0,1)$, then there exists a randomized polynomial-time algorithm solving the search version of $\mathrm{Gap}_{\epsilon_1}\mathrm{MCSP}$ for some $\epsilon_1 > 0$.*

For functions $f, g\colon \{0,1\}^n \to \{0,1\}$ and $\epsilon \in [0,1]$, we say that f is *ϵ-close* to g if $\mathrm{dist}(f,g) \le \epsilon$. The following is the main result of [17], which established a generic reduction from learning an ϵ-close function to natural properties.

Lemma 52 (Carmosino, Impagliazzo, Kabanets, and Kolokolova [17])**.** *For every $\ell \le n \in \mathbb{N}, \epsilon > 0$, there exists a "black-box generator" $G_{\ell,n,\epsilon}$ satisfying the following.*

- *$G_{\ell,n,\epsilon}$ maps a function $f\colon \{0,1\}^n \to \{0,1\}$ to a function $G^f_{\ell,n,\epsilon}\colon \{0,1\}^m \to \{0,1\}^{2^\ell}$ for some $m \in \mathbb{N}$, and*
- *$\mathsf{size}(G^f_{\ell,n,\epsilon}(z)) \le \mathsf{poly}(n, 1/\epsilon, \mathsf{size}(f))$ for all $z \in \{0,1\}^m$, where we regard $G^f_{\ell,n,\epsilon}(z)$ as a function on ℓ-bit inputs.*

Moreover, there exists a randomized polynomial-time oracle machine (a "reconstruction algorithm") satisfying the following specification.

Inputs: *Oracle access to a function $f\colon \{0,1\}^n \to \{0,1\}$, parameters $n, \epsilon^{-1}, 2^\ell \in \mathbb{N}$ represented in unary, and a circuit D on 2^ℓ-bit inputs.*

Promise: *We assume that D is a statistical test for $G^f_{\ell,n,\epsilon}$ with advantage δ_0. That is,*

$$\left| \Pr_{z \in_R \{0,1\}^m}\left[D(G^f_{\ell,n,\epsilon}(z)) = 1\right] - \Pr_{w \in_R \{0,1\}^{2^\ell}}\left[D(w) = 1\right] \right| \ge \delta_0,$$

for some universal constant $\delta_0 > 0$.

Output: *A circuit C that is ϵ-close to f. (In particular, the size of C is at most $\mathsf{poly}(n, \epsilon^{-1}, 2^\ell, |D|)$).*

Proof of Theorem 51. Suppose that the truth table of $f\colon \{0,1\}^n \to \{0,1\}$ is given as input. Let $u(\ell) := 2^{\epsilon_0 \ell}$ denote the usefulness parameter, and let $s := \mathsf{size}(f)$.

First, note that any circuit C that is ϵ-close to f can be converted into a circuit C' computing f *exactly* such that $|C'| \le |C| + \epsilon \cdot 2^n \cdot n + O(1)$. Indeed, since there are at most $\epsilon 2^n$ inputs on which f and C disagree, we can define a DNF formula φ with $\epsilon 2^n$ terms such that φ outputs 1 iff f and C disagree; then we may define $C'(x) := C(x) \oplus \varphi(x)$ so that $C'(x) = f(x)$ for every $x \in \{0,1\}^n$.

Therefore, the output of the reconstruction algorithm of Lemma 52 can be converted to a circuit C' computing f exactly such that $|C'| \le \mathsf{poly}(n, 1/\epsilon, 2^\ell, |D|) + \epsilon \cdot 2^n \cdot n$.

We now construct a statistical test for $G^f_{\ell,n,\epsilon}$ using a BPP-natural property and Adleman's trick [1] (for proving $\mathsf{BPP} \subseteq \mathsf{P/poly}$). Let $(L_{\mathrm{Y}}, L_{\mathrm{N}})$ be a BPP-natural property and M be a BPP algorithm solving $(L_{\mathrm{Y}}, L_{\mathrm{N}})$. Fix any $\ell \in \mathbb{N}$. By a standard error reduction for BPP, we may assume without loss of generality that the error probability of M is at most 2^{-2L} on any inputs of length $L := 2^\ell$. Pick a string r of length $\mathsf{poly}(\ell)$ uniformly at random, and hardwire r into M as the source of internal randomness. Then, by the union bound, with probability at least $1 - 2^{-L}$ over the choice of r, $M(\cdot; r)$ computes some natural property on input length L, i.e., $M(w; r) = 1$ iff $w \in L_{\mathrm{Y}}$, for every $w \in (L_{\mathrm{Y}} \cup L_{\mathrm{N}})^{=L}$. By a standard translation from a machine to a circuit, we convert $M(\cdot; r)$ to a circuit D_ℓ of size $\mathsf{poly}(\ell)$.

We claim that D_ℓ is a statistical test for $G^f_{\ell,n,\epsilon}$ if $\mathsf{size}(G^f_{\ell,n,\epsilon}(z)) \le u(\ell)$ for every z. Indeed, by the usefulness of natural properties, we have $G^f_{\ell,n,\epsilon}(z) \in L_{\mathrm{N}}$; thus $D_\ell(G^f_{\ell,n,\epsilon}(z)) = 0$. On the other hand, by the largeness of natural properties, we have $|(L_{\mathrm{Y}})^{=L}| \ge \delta_0 2^L$, and thus $\Pr_{w \in_R \{0,1\}^L}[D_\ell(w) = 1] \ge \delta_0$. Thus D_ℓ is a statistical test for $G^f_{\ell,n,\epsilon}$.

Here is an algorithm solving the search version of $\mathrm{Gap}_{\epsilon_1}\mathrm{MCSP}$. For every $\ell \in [n]$ and every $\epsilon^{-1} \in [2^n]$, run the reconstruction algorithm of Lemma 52 with inputs $f, n, \ell, 2^\ell$, and D_ℓ, and obtain a circuit C approximating f. Convert C to C' computing f exactly as explained above. Output the minimum circuit C' computing f found in this way.

It remains to claim that, for some choice of ℓ, ϵ, the reconstruction algorithm outputs a small circuit. Let $c > 0$ be a constant such that $G^f_{\ell,n,\epsilon}(z) \le (ns/\epsilon)^c$ and $|C'| \le (n2^\ell/\epsilon)^c + \epsilon 2^n n$ for all large n, ℓ, ϵ^{-1}. In order for D_ℓ to be a statistical test for $G^f_{\ell,n,\epsilon}$, we need $(ns/\epsilon)^c \le u(\ell) = 2^{\epsilon_0 \ell}$; thus we set $2^\ell := (ns/\epsilon)^{c/\epsilon_0}$. To make $|C'|$ small, we set $\epsilon := 2^{-\epsilon_1 n} s$ where $\epsilon_1 := (c + c^2/\epsilon_0 + 1)^{-1}$. Then we have $|C'| \le (n/\epsilon)^c (n 2^{\epsilon_1 n})^{c^2/\epsilon_0} + 2^{(1-\epsilon_1)n} ns \le n^{O(1)} 2^{(1-\epsilon_1)n} s \le 2^{(1-\epsilon_1/2)n} s$ for all large $n \in \mathbb{N}$. $\square$

Acknowledgment

The author would like to thank Eric Allender, Oded Goldreich, and the anonymous reviewers for helpful comments. This work was supported by ACT-I, JST and JSPS KAKENHI Grant Numbers JP16J06743.

References

[1] Leonard M. Adleman. Two theorems on random polynomial time. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 75–83, 1978.

[2] Dorit Aharonov and Oded Regev. Lattice problems in NP $\cap$ coNP. *J. ACM*, 52(5):749–765, 2005.

[3] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Symposium on the Theory of Computing (STOC)*, pages 99–108, 1996.

[4] 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.

[5] Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006.

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

[7] Eric Allender, Joshua A. Grochow, Dieter van Melkebeek, Cristopher Moore, and Andrew Morgan. Minimum circuit size, graph isomorphism, and related problems. In *Proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*, pages 20:1–20:20, 2018.

[8] Eric Allender, Lisa Hellerstein, Paul McCabe, Toniann Pitassi, and Michael E. Saks. Minimizing disjunctive normal form formulas and AC^0 circuits given a truth table. *SIAM J. Comput.*, 38(1):63–84, 2008.

[9] Eric Allender and Shuichi Hirahara. New insights on the (non-)hardness of circuit minimization and related problems. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 54:1–54:14, 2017.

[10] Eric Allender, Dhiraj Holden, and Valentine Kabanets. The minimum oracle circuit size problem. *Computational Complexity*, 26(2):469–496, 2017.

[11] Eric Allender, Michal Koucký, Detlef Ronneburger, and Sambuddha Roy. The pervasive reach of resource-bounded kolmogorov complexity in computational complexity theory. *J. Comput. Syst. Sci.*, 77(1):14–40, 2011.

[12] 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.

[13] 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.

[14] Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Foundations and Trends in Theoretical Computer Science*, 2(1), 2006.

[15] Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for NP problems. *SIAM J. Comput.*, 36(4):1119–1159, 2006.

[16] Peter Bürgisser, Oded Goldreich, Madhu Sudan, and Salil Vadhan. Complexity theory. *Oberwolfach Reports*, 12(4):3049–3099, 2016.

[17] 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.

[18] Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 534–543, 2002.

[19] Uriel Feige, Jeong Han Kim, and Eran Ofek. Witnesses for non-satisfiability of dense random 3cnf formulas. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 497–508, 2006.

[20] Uriel Feige and Eran Ofek. Easily refutable subformulas of large random 3cnf formulas. *Theory of Computing*, 3(1):25–43, 2007.

[21] Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM J. Comput.*, 22(5):994–1005, 1993.

[22] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[23] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. If NP languages are hard on the worst-case, then it is easy to find their hard instances. *Computational Complexity*, 16(4):412–441, 2007.

[24] 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.

[25] Ishay Haviv and Oded Regev. Tensor-based hardness of the shortest vector problem to within almost polynomial factors. *Theory of Computing*, 8(1):513–531, 2012.

[26] Shuichi Hirahara, Igor Carboni Oliveira, and Rahul Santhanam. Np-hardness of minimum circuit size problem for OR-AND-MOD circuits. In *Proceedings of the Computational Complexity Conference (CCC)*, pages 5:1–5:31, 2018.

[27] 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.

[28] Shuichi Hirahara and Osamu Watanabe. Limits of minimum circuit size problem as oracle. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 18:1–18:20, 2016.

[29] Shuichi Hirahara and Osamu Watanabe. Simulating nonadaptive reductions to natural proofs by constant-round interactive proofs. Manuscript, 2018.

[30] John M. Hitchcock and Aduri Pavan. On the NP-completeness of the minimum circuit size problem. In *Proceedings of the Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 236–245, 2015.

[31] Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the Structure in Complexity Theory Conference*, pages 134–147, 1995.

[32] Russell Impagliazzo. Relativized separations of worst-case and average-case complexities for NP. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 104–114, 2011.

[33] Russell Impagliazzo, Valentine Kabanets, and Ilya Volkovich. The power of natural properties as oracles. In *Proceedings of the Computational Complexity Conference (CCC)*, pages 7:1–7:20, 2018.

[34] 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.

[35] Russell Impagliazzo and Avi Wigderson. Randomness vs time: Derandomization under a uniform assumption. *J. Comput. Syst. Sci.*, 63(4):672–688, 2001.

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

[37] Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM J. Comput.*, 31(5):1501–1526, 2002.

[38] Ker-I Ko. On the complexity of learning minimum time-bounded turing machines. *SIAM J. Comput.*, 20(5):962–986, 1991.

[39] Leonid A. Levin. Average case complete problems. *SIAM J. Comput.*, 15(1):285–286, 1986.

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

[41] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007.

[42] Cody D. Murray and R. Ryan Williams. On the (non) NP-hardness of computing circuit complexity. *Theory of Computing*, 13(1):1–22, 2017.

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

[44] Igor Carboni Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *Proceedings of the Computational Complexity Conference (CCC)*, pages 18:1–18:49, 2017.

[45] Ran Raz, Omer Reingold, and Salil P. Vadhan. Extracting all the randomness and reducing the error in Trevisan's extractors. *J. Comput. Syst. Sci.*, 65(1):97–128, 2002.

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

[47] Steven Rudich. Super-bits, demi-bits, and NP/qpoly-natural proofs. In *Proceedings of the Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 85–93, 1997.

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

[49] Boris A. Trakhtenbrot. A survey of russian approaches to perebor (brute-force searches) algorithms. *IEEE Annals of the History of Computing*, 6(4):384–400, 1984.

[50] Luca Trevisan. Extractors and pseudorandom generators. *J. ACM*, 48(4):860–879, 2001.

[51] Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007.

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

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

[54] 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.