Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 138 (2018)

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

Shuichi Hirahara[*]
The University of Tokyo
hirahara@is.s.u-tokyo.ac.jp

February 28, 2019

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 conjectured to be outside coNP 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. We observe that the approximation version of MINKT is Random 3SAT-hard, and more generally it is harder than avoiding any polynomial-time computable hitting set generator that extends its seed of length n by $\widetilde{\omega}(\sqrt{n})$, which provides strong evidence that the approximation problem is outside coNP and thus our reductions are non-black-box. Our reduction can be derandomized at the cost of the quality of the approximation. 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.

Our results can be seen as a new approach for excluding Heuristica. In particular, proving NP-hardness of the approximation versions of MINKT or the Minimum Circuit Size Problem (MCSP) is sufficient for establishing an equivalence between the worst-case and average-case hardness of NP.

1 Introduction

The security of cryptographic primitives such as public-key cryptosystems is based on some hardness assumption of NP. Indeed, if $\mathsf{P} = \mathsf{NP}$, then essentially all cryptographic primitives can be broken efficiently. This is intuitively because NP is the complexity class of problems whose Yes instances have a certificate that is efficiently checkable; in order to build a meaningful cryptographic primitive, it is required that a legitimate user who has a secret key (i.e., a certificate) must be verified efficiently. More formally, the existence of a *one-way function* (OWF) is often considered as a minimal complexity assumption to build cryptography [IL89]. Roughly speaking, a one-way function is a function f such that f is easy to compute but no efficient adversary can invert f on

[*]Supported by ACT-I, JST and JSPS KAKENHI Grant Numbers JP16J06743.

ISSN 1433-8092

average; it is easy to see that any one-way function can be inverted with NP oracles. Thus the security of cryptographic primitives must be based on intractability of some NP problem, and hence proving $\mathsf{P} \neq \mathsf{NP}$ is the first step towards the construction of secure cryptography.

However, there is a huge gap between $\mathsf{P} \neq \mathsf{NP}$ and the existence of public-key cryptography. This gap was already discussed in the seminal work of Diffie and Hellman [DH76], which introduced public-key cryptography. The main problem is that traditional complexity classes such as P and NP measure the performance of an algorithm with respect to the *worst-case* input; therefore, the statement $\mathsf{P} \neq \mathsf{NP}$ only tells us that there exists *some* hard input on which a polynomial-time machine cannot solve some NP problem; however, it does not tell us how to generate such a hard input efficiently. In contrast, for the purpose of cryptography, we need to efficiently generate a secret key randomly so that an adversary cannot find the secret key in a reasonable amount of time. Thus we need to understand the *average-case complexity* of NP: that is, how much time on average does it take to compute NP problems on efficiently generated random inputs?

1.1 Impagliazzo's Five Worlds

In the influential survey on average-case complexity of Impagliazzo [Imp95], the gap between $\mathsf{P} \neq \mathsf{NP}$, average-case complexity of NP, and the existence of cryptographic primitives was clearly addressed. He explored five possible scenarios that are consistent with our current knowledge on complexity theory, and named each possible world as follows: Algorithmica (where NP is easy on the worst-case; e.g., $\mathsf{P} = \mathsf{NP}$), Heuristica (where NP is 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). The five worlds are classified according to the four central open questions in complexity theory, and exactly one of the possible 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 central 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.

Since all of these questions are the central open questions in complexity theory and cryptography, a lot of work has been done for each open question in order to understand why current proof techniques are not capable of resolving the questions. For example, in order to resolve $\mathsf{P} \neq \mathsf{NP}$ (or, in other words, to exclude Algorithmica from the possible worlds), we need to develop a new proof technique that overcomes the *relativization* barrier [BGS75], the *algebrization* barrier [AW09], and the *natural proof* barrier [RR97] simultaneously. Similarly, in order to exclude Heuristica (e.g., $\mathsf{P} \neq \mathsf{NP} \Longrightarrow \mathsf{DistNP} \not\subseteq \mathsf{AvgP}$), we need to develop a new proof technique that overcomes a relativization barrier [Imp11] and *limits of black-box reductions* [FF93, BT06b]. The main contribution of this paper is to overcome the limits of black-box reductions, by presenting the first non-black-box worst-case to average-case reductions. We achieve this by investigating the complexity of problems of compressing a given string by certain types of algorithms, namely MCSP and MINKT.

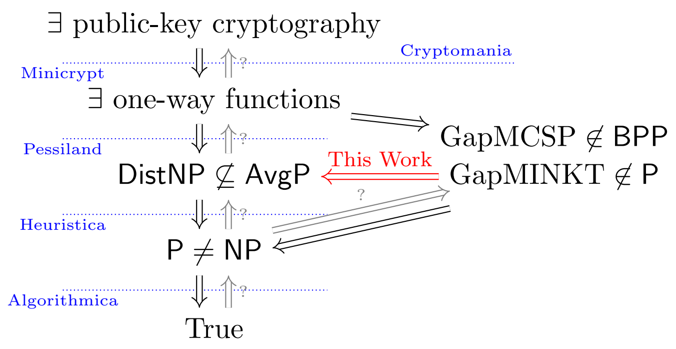

Figure 1: Impagliazzo's five worlds and the main contribution of this work. In particular, proving NP-hardness of GapMINKT (i.e., $\mathsf{P} \neq \mathsf{NP} \overset{?}{\Rightarrow} \text{GapMINKT} \notin \mathsf{P}$) is sufficient for excluding Heuristica (i.e., $\mathsf{P} \neq \mathsf{NP} \overset{?}{\Rightarrow} \mathsf{DistNP} \not\subseteq \mathsf{AvgP}$).

1.2 Minimum Circuit Size Problem (MCSP) and Its Variants

The *Minimum Circuit Size Problem* (MCSP [KC00]) asks for compressing a given input f into the truth table of a small circuit. More formally, 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 $s \in \mathbb{N}$, it asks whether there exists a circuit of size at most s computing f. Similarly, MINKT (Minimum Kolmogorov Time-bounded Complexity [Ko91]) 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. We denote by GapMCSP and GapMINKT approximation versions of these problems, respectively. (There is another variant called MKTP [ABK$^+$06, AHK17], 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, neither NP-completeness proof nor 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 [Tra84]. Indeed, it is reported in [AKRR11] that Levin delayed his publication on the NP-completeness of SAT [Lev73] because he wanted to prove a similar result for MCSP. It is thus a long-standing open problem in complexity theory whether MCSP (or GapMINKT) is NP-complete or not. The open question is depicted in Figure 1 as the implication "$\mathsf{NP} \neq \mathsf{P} \overset{?}{\Rightarrow} \text{GapMINKT} \notin \mathsf{P}$."[1]

A fundamental relationship between cryptography and MCSP was discovered in the celebrated natural proof framework of Razborov and Rudich [RR97], based on which Kabanets and Cai [KC00] reawakened interest in MCSP. Since then many efforts have been made to understand the complexity of MCSP (e.g., [ABK$^+$06, AHM$^+$08, AKRR11, AD17, MW17, HP15, HW16, CIKK16, OS17, HS17, AH17, AGvM$^+$18, IKV18, HOS18]). In particular, any one-way function can be inverted if GapMCSP is in BPP [ABK$^+$06]. This corresponds to the implication "$\exists$ one-way functions $\Rightarrow$ GapMCSP $\notin \mathsf{BPP}$."

In this paper, we show that worst-case hardness of GapMCSP or GapMINKT implies average-case hardness of NP. In particular, our results can be seen as a new approach for excluding

[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.

Heuristica: Proving NP-completeness of GapMINKT is sufficient for excluding Heuristica, i.e., a world where $\mathsf{P} \neq \mathsf{NP}$ and $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ (see Figure 1). The latter is a central open question in the theory of average-case complexity, as we review next.

1.3 Average-case Complexity

Average-case complexity, pioneered by Levin [Lev86], 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 $\mathcal{D} = \{\mathcal{D}_m\}_{m \in \mathbb{N}}$, where m means the size of instances. 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 [BT06a] 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. [FF93, BFNW93, STV01]).

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 [Ajt96], it is shown that an approximation 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 $\widetilde{O}(n)$ [MR07]. 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 $\mathsf{NP} \cap \mathsf{coNP}$ [GG00, AR05]. Some NP-hardness is known for an approximation factor of $n^{O(1/\log\log n)}$ [HR12].

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

The proof techniques of the previous work mentioned above are *(black-box) reductions*. Namely, for a worst-case problem L and a distributional problem $(L', \mathcal{D})$, we design an efficient algorithm R such that, given oracle access to an errorless heuristic algorithm T that solves $(L', \mathcal{D})$, R solves L. The correctness of a reduction is often established no matter how T is complex – such a reduction is called *black-box* in the sense that T is regarded as a (potentially inefficient) black-box oracle, and we just care about the input-output behavior of T.

Such a black-box reduction technique turned out to have a certain limit. Building on Feigenbaum and Fortnow [FF93], Bogdanov and Trevisan [BT06b] 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 "/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 [AGGM06, BB15] or avoiding hitting set generators [HW19]. 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 [GST07] 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 $\mathcal{D}^{\mathrm{KT}}$, 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.

2 Overview of Our Results

The main contribution of this paper is to present the first *non-black-box* worst-case to average-case reductions from problems conjectured to be outside $\mathsf{NP} \cap \mathsf{coNP}$ to distributional NP problems. As a consequence of our reductions, we improve our understanding around Heuristica as shown in Figure 2.

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 $r \colon \mathbb{N} \to \mathbb{N}$.

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

Theorem 4.21 (Main). *Let $r \colon \mathbb{N} \to \mathbb{N}$ be any function such that for some constant $c > 0$, for all large $n \in \mathbb{N}$, $n - c\sqrt{n}\log n \leq r(n) < n$. Assume that $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}}) \in \mathsf{Avg}_{1/6m}\mathsf{P}$. 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*

[2] Here we implicitly used a popular conjecture that $\mathsf{AM} = \mathsf{NP}$ [KvM02]. We also mention that an *adaptive* black-box reduction could be used to overcome the barriers.

$$\begin{tikzcd}
& \mathsf{SZK} \not\subseteq \mathsf{BPP} \arrow[d, Rightarrow, "\text{[Ost91]}"] & \\
\exists\,\mathrm{OWF} \arrow[r, Rightarrow] \arrow[d, Leftrightarrow, "\text{[HILL99]}"] & \exists\,\mathrm{AIOWF} \arrow[r, Rightarrow, "\text{[AH17]}"'] & \mathrm{Gap}_{\epsilon}\mathrm{MCSP} \notin \mathsf{BPP}\ (\forall \epsilon > 0) \arrow[d, Rightarrow, "\text{Theorem 6.7}"] \\
\exists\,\mathrm{PRG} \arrow[rr, Rightarrow] \arrow[d, Rightarrow] & & \exists\,\mathrm{HSG} \arrow[dd, Rightarrow, "\text{Corollary 5.3}"] \\
\mathsf{DistNP} \not\subseteq \mathsf{HeurP} \arrow[d, Rightarrow] & & \\
\mathsf{DistNP} \not\subseteq \mathsf{AvgP} \arrow[d, Rightarrow] & & \mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \notin \mathsf{P} \arrow[ll, Rightarrow, "\text{Corollary 4.23}"'] \\
\mathsf{P} \neq \mathsf{NP} & &
\end{tikzcd}$$

Figure 2: An improved landscape around Heuristica. By "$\exists\,\mathrm{HSG}$", we mean that there exists an efficiently computable hitting set generator $G = \{G_n : \{0,1\}^{s(n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ secure against either P or BPP, where the seed length $s(n)$ satisfies $n^{\Omega(1)} \le s(n) \le n - \widetilde{\omega}(\sqrt{n})$.

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, with high probability.

At a high level, the reason why Theorem 4.21 is non-black-box is that the compressed program M incorporates an errorless heuristic algorithm T for $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}})$. In particular, M can be decoded in polynomial time if $T \in \mathsf{P}$, but in the other case, M may not be decoded efficiently.

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 4.21, we prove the following relationship between the worst-case problem $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ and the complexity of DistNP.

Corollary 4.23. *If $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ then $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in \text{Promise-}\mathsf{P}$ for some $\sigma(n,s) = s + O\big((\log n)\sqrt{s} + (\log n)^2\big)$ and some polynomial $\tau(n,t)$.*

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 6.7. *The following are equivalent.*

1. $\mathrm{Gap}_{\epsilon}\mathrm{MCSP} \in \text{Promise-}\mathsf{BPP}$ *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* $\epsilon \in (0,1)$.

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

We will observe that a natural property useful against $\mathsf{SIZE}(2^{\epsilon n})$ is an adversary for a specific hitting set generator $G^{\mathsf{int}, n^{\epsilon}}$ of seed length $\widetilde{O}(n^{\epsilon})$ (defined as an interpreter of a circuit; see Definition 5.4 and Definition 5.5). In particular, it follows from Theorem 6.7 that $\mathrm{Gap}_{\epsilon}\mathrm{MCSP} \notin \text{Promise-}\mathsf{BPP}$ for every $\epsilon > 0$ implies the existence of a hitting set generator $G^{\mathsf{int}, n^{\epsilon}}$ secure against BPP algorithms for every $\epsilon \in (0, 1)$, as depicted in Figure 2.

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 [HS17]; 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, 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.

2.1 Hardness of GapMINKT

We argue that our techniques are inherently non-black-box. If Theorem 4.21 were established via a nonadaptive black-box worst-case to average-case reduction, then by using the techniques of Bogdanov and Trevisan [BT06b], we would obtain $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in \mathsf{coNP}/\mathsf{poly}$. This is unlikely, as we discuss below. (In fact, our non-black-box reductions can be regarded as a nonadaptive reduction to avoiding a hitting set generator; thus, the advice "$/\mathsf{poly}$" is not indispensable [HW19].)

Unfortunately, basing hardness of MCSP or MINKT on *worst-case hardness assumptions* is a very challenging task. The current best worst-case hardness result for MCSP is SZK-hardness, which is proved by inverting auxiliary-input one-way functions (Allender and Das [AD17]). Here we say that an auxiliary-input function $f_{(\text{-})}(\text{-})$ is *inverted* by a randomized algorithm A if, for every auxiliary input $x \in \{0, 1\}^*$, $A(x, f_x(y))$ returns an element of $f_x^{-1}(f_x(y))$ with high probability over the choice of coin flips of A and a string y chosen from the uniform distribution; f is said to be an *auxiliary-input one-way function* (AIOWF) if there is no randomized polynomial-time algorithm that inverts f. It was shown by Ostrovsky [Ost91] that $\mathsf{SZK} \not\subseteq \mathsf{BPP}$ implies the existence of an auxiliary-input one-way function (see also [Vad06, Theorem 7.5]). Moreover, building on [RR97, GGM86, HILL99, ABK+06], it was shown in [AH17] that an auxiliary-input one-way function can be inverted in polynomial time with oracle access to $\mathrm{Gap}_{\epsilon}\mathrm{MCSP}$, for every constant $\epsilon > 0$. We summarize these relationships in Figure 2.

The SZK-hardness sketched above 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 cannot be used to base hardness of MCSP on any problem beyond $\mathsf{AM} \cap \mathsf{coAM}$ [HW16]. 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 hardness assumptions*. Hirahara and Santhanam [HS17] showed that MKTP is Random 3SAT-hard, 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 avoid any hitting set generator.

Corollary 5.3. *Let σ, τ be the parameters as in Theorem 4.21. Any efficiently computable hitting set generator $G = \{G_n : \{0,1\}^{n-\widetilde{\omega}(\sqrt{n})} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ is not secure against a polynomial-time algorithm with oracle access to $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$.*

This is because any range of an efficiently computable hitting set generator is not random in the sense of time-bounded Kolmogorov complexity; thus, to test whether x is in the range of G, 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. Based on some average-case hardness assumption about the subset sum problem, Rudich [Rud97] conjectured that there is no $\mathsf{NP}/\mathsf{poly}$-natural property useful against P/poly. In particular, under his conjecture, we have $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \notin \mathsf{coNP}/\mathsf{poly}$ (and $\mathrm{Gap}_{\epsilon}\mathrm{MCSP} \notin \mathsf{coNP}/\mathsf{poly}$).

More significantly, we observe that 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., [Fei02, FO07, FKO06]). This the problem of checking the satisfiability of a 3CNF formula randomly generated by choosing m clauses uniformly at random from all the possible width-3 clauses on n variables. The best coNP algorithm solving Random 3SAT is the algorithm given by Feige, Kim and Ofek [FKO06], which works when $m > O(n^{7/5})$; this is better than the best deterministic algorithm, which works when $m > O(n^{3/2})$ [FO07].

We show that if $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in \mathsf{coNP}$, there is a much better algorithm than [FKO06]: 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 a coNP algorithm with probability $1 - 2^{-\Omega(n)}$. Ryan O'Donnell (cf. [HS17, BGSV16]) conjectured that there is no coNP algorithm solving Random 3SAT with $m = \Delta n$ clauses for a sufficiently large constant Δ. Thus under his conjecture, we have $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \notin \mathsf{coNP}$.

2.2 Perspective: An Approach Towards Excluding Heuristica

We propose a research program towards excluding Heuristica through the lens of MCSP or MINKT. Note that if $\mathsf{NP} \leq_T^{\mathsf{P}} \mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ then it follows from Corollary 4.23 that Heuristica does not exist, in the sense that $\mathsf{P} \neq \mathsf{NP}$ implies $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$.

Unfortunately, there are still several obstacles we need to overcome in order for this research program to be completed. Although our results overcome the limits of black-box reductions, most of our results do *relativize*. (One potential exception is Corollary 4.23, where we use a nonrelativizing proof technique of [BFP05].) And there is a relativization barrier for excluding Heuristica: Impagliazzo [Imp11] 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 4.21 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 [Ko91] 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. [KC00, ABK+06, AHK17, MW17, HW16, HP15, AH17]) 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 [MW17] showed that MCSP is provably not NP-hard under some sublinear time reductions; similarly, NP-hardness of GapMCSP under polynomial-time Turing reductions implies $\mathsf{EXP} \neq \mathsf{ZPP}$ [HW16].

However, few is known for weaker reducibility notions such as $\mathsf{NP} \cap \mathsf{coNP}$ reductions. We suggest that the following is an interesting research direction.

Open Question 2.1. Prove the following (or explain why it is difficult to resolve): Let σ, τ be arbitrary parameters as in Theorem 4.21. $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ is NP-hard under $\mathsf{coNP}/\mathsf{poly}$-Turing reductions. That is, $\mathsf{NP} \subseteq \mathsf{coNP}^A/\mathsf{poly}$ for any oracle A that satisfies the promise of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$.

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 [Ko91] also speculated that MINKT might be NP-complete under $\mathsf{NP} \cap \mathsf{coNP}$ reductions. We mention that there are nonrelativizing proof techniques for proving PSPACE-completeness of a space-bounded version of MINKT under ZPP-Turing reductions and EXP-completeness of an exponential-time version of MINKT under $\mathsf{NP} \cap \mathsf{coNP}$-Turing reductions (cf. [ABK+06]). In particular, under the (unlikely) assumption that $\mathsf{PSPACE} \subseteq \mathsf{P}/\mathsf{poly}$, MCSP is indeed NP-hard under ZPP-Turing reductions (cf. [IKV18]).

A positive answer to Open Question 2.1 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 $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$.

2.3 Our Techniques

We outline the proof of Theorem 4.21 below. The basic idea is to make use of the hardness versus randomness framework based on the Nisan-Wigderson pseudorandom generator [NW94]. Specifically, based on a hard function f, they constructed a pseudorandom generator NW^f. The security of the pseudorandom generator is proved by the following reduction: Given any statistical test T that distinguishes NW^f from the uniform distribution, one can construct a small T-oracle circuit that approximates f. Such a security proof turns out to be quite fruitful not only for derandomization [KvM02, IW01, TV07], but also for Trevisan's extractor [Tre01], investigating the power of Kolmogorov-random strings [ABK+06], the language compression problem [BLvM05], and the generic connection between learning and natural proof [CIKK16], to mention a few. Our proofs are also based on the security proof, and we combine it with a statistical test that can be constructed from an errorless heuristic algorithm for MINKT.

A Statistical Test for Every Pseudorandom Generator. Specifically, we observe that an errorless heuristic algorithm for $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}})$ implies the existence of a statistical test T that distinguishes every pseudorandom generator from the uniform distribution. A crucial observation here 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 $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}})$ must succeed on a large fraction of No instances. This gives rise to an algorithm T that rejects most inputs and accepts every Yes instance, and it can be shown that T is a statistical test for any hitting set generator. This observation was inspired by the work of Hirahara and Santhanam [HS17], which showed the equivalence between natural properties and an errorless heuristic algorithm for MCSP.

In particular, T distinguishes *any* candidate pseudorandom generator from the uniform distribution. In what follows, we explain the ideas of constructing a candidate pseudorandom generator

G^x based on a given string $x \in \{0,1\}^*$, and outline how to convert a statistical test T for G^x into a short program that describes x. We emphasize that in order to compress x into a short program using T, it is important that a statistical test T does not depend on a pseudorandom generator G^x.

Converting a Statistical Test to Short Descriptions. Let f be a hard function; then a function G_1^f defined as $G_1^f(z) := (z, f(z))$ is known to be a secure pseudorandom generator that extends its seed by one bit. This is shown by using the connection between next-bit unpredictability and a pseudorandom generator: Let T be a statistical test that distinguishes $G_1^f(z)$ from the uniform distribution, that is, $\mathbb{E}_{z \in_R \{0,1\}^\ell, b \in_R \{0,1\}}[T(G_1^f(z)) - T(z,b)] \geq \epsilon$; Then a randomized T-oracle circuit P^T defined as $P^T(z) := T(z,b) \oplus b \oplus 1$ where $b \in_R \{0,1\}$ predicts $f(z)$ with probability at least $\frac{1}{2} + \epsilon$. This means that a $(\frac{1}{2} + \epsilon)$-fraction of f can be described by a short program with oracle access to T.

Given a string x which we would like to compress, we regard x of length 2^ℓ as the truth table of a function $x \colon \{0,1\}^\ell \to \{0,1\}$. From the argument above, we can convert any statistical test T for G_1^x into a short program that describes a $(\frac{1}{2} + \epsilon)$-fraction of x. Using a list-decodable error-correcting code Enc, we can convert any statistical test T for $G_2^x := G_1^{\mathrm{Enc}(x)}$ into a short description for x. Thus it remains to argue that G_2^x can be distinguished by using an errorless heuristic algorithm for $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}})$.

Note that G_2^x extends its seed by only one bit, which is too small to detect. Thus we extend the output length of G_2^x by using, e.g., the direct product construction: Define $G_3^x(z^1, \cdots, z^m) := G_2^x(z^1) \cdots G_2^x(z^m)$ for a parameter $m \in \mathbb{N}$ and $(z^1, \cdots, z^m) \in (\{0,1\}^\ell)^m$. Since the Kolmogorov complexity of $G_3^x(z^1, \cdots, z^m)$ is approximately less than $m\ell + \mathrm{K}_t(x)$, by measuring Kolmogorov complexity, G_3^x can be distinguished from the uniform distribution $w \in_R \{0,1\}^{m\ell+m}$ if $m\ell + \mathrm{K}_t(x) \ll m\ell + m$. Taking $m \approx \mathrm{K}_t(x)$, we can construct a statistical test T that distinguishes G_3^x. By a standard hybrid argument, the statistical test can be converted into a T-oracle program of size $O(m\ell)$ that distinguishes G_2^x (where $O(m\ell)$ bits are used to specify random bits that maximize the advantage of T), and thus we obtain a T-oracle program of length $O(\mathrm{K}_t(x) \log |x|)$ describing x.

In order to reduce the description length, we make use of the Nisan-Wigderson pseudorandom generator [NW94] (and an improved construction of combinatorial designs due to Raz, Reingold and Vadhan [RRV02]) in the actual construction. Specifically, for a seed z, we generate m inputs $z_{S_1}, \cdots, z_{S_m}$ by using a nearly disjoint set family $\{S_1, \cdots, S_m\}$, and define $\mathrm{NW}^{\mathrm{Enc}(x)}(z) := \mathrm{Enc}(x)(z_{S_1}) \cdots \mathrm{Enc}(x)(z_{S_m})$, where we identify $\mathrm{Enc}(x) \in \{0,1\}^{2^\ell}$ with a Boolean function $\mathrm{Enc}(x) \colon \{0,1\}^\ell \to \{0,1\}$. A standard security proof of the Nisan-Wigderson pseudorandom generator enables us to convert a statistical test T for $\mathrm{NW}^{\mathrm{Enc}(x)}$ into a T-oracle program of size $\mathrm{K}_t(x) + \widetilde{O}(\sqrt{\mathrm{K}_t(x)})$. (We mention that our analysis and the quality of the approximation is similar to the work of Buhrman, Lee and van Melkebeek [BLvM05] on the language compression problem.)

As a consequence, we obtain an efficient algorithm that, on input x, outputs a short T-oracle program d describing x. Since T can be accepted by some polynomial-time algorithm (that comes from the errorless heuristic algorithm for $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}})$), 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. More importantly, in order to prove the correctness of our reductions, we crucially exploit the fact that the oracle T is in P: The compressed program can be decoded in polynomial time when $T \in \mathsf{P}$, but in the other case, the program may not be decoded efficiently.

We note that the algorithm outlined above is a one-sided-error randomized algorithm. However, using a proof idea of the fact that $\mathsf{BPP} = \mathsf{ZPP}$ if $\mathsf{DistNP} \subseteq \mathsf{AvgZPP}$ [Imp95], we can make the randomized algorithm zero-error. This enables us to establish Theorem 4.21. Our reductions can be completely derandomized either by using the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ (under which Buhrman, Fortnow and Pavan [BFP05] constructed a pseudorandom generator), or at the cost of the quality of the approximation (cf. Theorem 4.24).

2.3.1 Reductions for MCSP

To prove an 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 [CIKK16] 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 remark that $\mathrm{Amp}(f)$ can be seen as a locally encodable and locally decodable error-correcting code [Zim05].) We observe that their approximately learning is enough to achieve the approximation factor stated in Theorem 6.7. By combining their results with the equivalence between a natural property and an errorless heuristic algorithm for MCSP [HS17], we obtain a search to average-case reduction for GapMCSP.

Organization. In Section 3, we review background on Kolmogorov complexity. Then in Section 4, we give a search to average-case reduction for MINKT and prove Theorem 4.21. In Section 5, we present evidence against $\mathrm{MINKT} \in \mathsf{coNP}$. Section 6 is devoted to proving Theorem 6.7.

3 Preliminaries

We introduce several notations used throughout this paper. 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 $x \in \{0,1\}^*$. 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 $f\colon \{0,1\}^\ell \to \{0,1\}$, the *truth table* of a function is $f(z_1)\cdots f(z_{2^\ell})$, where $z_1, \dots, z_{2^\ell} \in \{0,1\}^\ell$ are all the strings of length ℓ in lexicographic order. We will sometimes identify a function f with its truth table.

Promise Problem. A *promise problem* is a pair $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$ of languages $\Pi_{\text{YES}}, \Pi_{\text{NO}} \subseteq \{0,1\}^*$ such that $\Pi_{\text{YES}} \cap \Pi_{\text{NO}} = \varnothing$, where Π_{YES} and Π_{NO} are regarded as the set of YES and NO instances, respectively. If $\Pi_{\text{YES}} = \{0,1\}^* \setminus \Pi_{\text{NO}}$, we identify $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$ with the language $\Pi_{\text{YES}} \subseteq \{0,1\}^*$. We say that a language A *satisfies* the promise of $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$ if $\Pi_{\text{YES}} \subseteq A \subseteq \{0,1\}^* \setminus \Pi_{\text{NO}}$. 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 the 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 3.1 (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 3.2 (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 under the oracle 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 3.3 (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 $\mathrm{K}_{t,\delta}^A(f)$ as the minimum length of a string d such that $U^A(d)$ outputs the truth table of some function g of length 2^ℓ within t steps and $\mathrm{dist}(f,g) \le 1/2 - \delta$.*

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

Definition 3.4 (Ko [Ko91])**.** *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 3.5. *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 4.21, 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 an approximation version of MINKT, parameterized by σ and τ.

Definition 3.6 (Approximation version of MINKT)**.** *Let $\sigma, \tau \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be any functions such that $\sigma(n, s) \geq s$ and $\tau(n, t) \geq t$ 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 $\mathrm{K}_t(x) \leq s$.*
- 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 3.7 (Search version of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$)**.** *For any functions σ, τ as in Definition 3.6, 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 $t' \geq \tau(|x|, t)$.*

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 $A(x, 1^t) \neq \bot$, 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 3.8 (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 $|d| \leq \sigma(|x|, s)$ and $U(d)$ outputs x in $\tau(|x|, t)$ steps.

We claim the correctness of this algorithm. If $(x, 1^t, 1^s)$ is a 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 $\bot$; that is, $U(d)$ outputs x in t' steps, and $|d| \leq \sigma(|x|, \mathrm{K}_t(x)) \leq \sigma(|x|, s)$. 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 $\bot$. $\square$

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

Lemma 3.9. *Let $T \in \mathsf{P}$. Then there exists some polynomial p such that $\mathrm{K}_{t'}(x) \leq \mathrm{K}^T_t(x) + O(1)$ for any $x \in \{0, 1\}^*$ and any t, t' such that $t' \geq p(t)$. 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 4.21). 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 $\mathrm{K}_{t'}(x) \preceq s + O(|M|)$. ☐

4 Worst-Case to Average-Case Reduction for MINKT

In this section, we present an efficient algorithm that outputs a certificate for GapMINKT under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$. The proof consists of two parts. In Subsection 4.1, we show the existence of such an efficient algorithm, assuming that the algorithm is given access to an oracle that accepts some dense subset of random strings. The existence of such an oracle will be justified in Subsection 4.2 under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$.

4.1 Short Certificate Under a Dense Subset of Random Strings

A string $x \in \{0,1\}^*$ is said to be *random* if x does not have a shorter description than itself. More generally:

Definition 4.1 (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 4.2 (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 $|A| \ge 2^m\delta$.

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 distinguish the Nisan-Wigderson generator from the uniform distribution.

We proceed to define the Nisan-Wigderson generator NW^f. Originally, Nisan and Wigderson [NW94] 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 [RRV02], a weaker notion is sufficient for a security proof of the Nisan-Wigderson generator. However, our situations are different from theirs. We thus modify the definition so that it captures the quality of the approximation.

Definition 4.3. *For a family $\mathcal{S} = (S_1, \dots, S_m)$ of ℓ-sized subsets of $[d]$, we say that $\mathcal{S}$ is of* quality *ρ if, for every $i \in [m]$,*

$$\sum_{j=1}^{i-1} 2^{|S_i \cap S_j|} + m - i + d \le \rho.$$

For example, if $\mathcal{S}$ is a family of m disjoint subsets of size ℓ, then its quality is $\max_i \sum_{j=1}^{i-1} 2^{|S_i \cap S_j|} + m - i + d \le m + m\ell$, since the size of the universe is $d = m\ell$. This enables us to obtain an approximation factor of $\ell = O(\log |x|)$. In order to achieve better quality, we use the "weak design" constructed in [RRV02].

Lemma 4.4. *Let $\rho(m, \ell, d) := \exp(\ell^2/d) \cdot m + d$. For any $m, \ell, d \in \mathbb{N}$ such that $d/\ell \in \mathbb{N}$, there exists a ρ-quality family $\mathcal{S}_{m,\ell,d} = (S_1, \dots, S_m) \subseteq \binom{[d]}{\ell}$, where $\rho = \rho(m, \ell, d)$. Moreover, the family $\mathcal{S}_{m,\ell,d}$ can be constructed by a deterministic algorithm in time $\mathsf{poly}(m, d)$.*

Proof. Raz, Reingold and Vadhan [RRV02, Lemma 15] 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 $\sum_{j=1}^{i-1} 2^{|S_i \cap S_j|} + m - i + d \le \exp(\ell^2/d) \cdot m + d$ for every $i \in [m]$. $\square$

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 $z_{i_1} \cdots z_{i_\ell}$. We define the Nisan-Wigderson generator based on $\mathcal{S}_{m,\ell,d}$. In what follows, we treat d/ℓ as if it is a variable in order to avoid introducing a new variable.

Definition 4.5 (Nisan-Wigderson generator [NW94])**.** *For a function $f\colon \{0,1\}^\ell \to \{0,1\}$ and parameters $m, \ell, d/\ell \in \mathbb{N}$, we 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 [NW94] showed that if 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 [IW01], such small circuits can be constructed efficiently. We now make use of these facts to obtain a short description for f. Our proof is reminiscent of the proof of Trevisan's extractor [Tre01], but we need to argue the efficiency.

Lemma 4.6. *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,*

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

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

Proof. We first prove $\mathrm{K}^T_{t,\,\delta/m}(f) \leq \rho(m,\ell,d) + O(\log(md))$. 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 [NW94]. 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 $\mathbb{E}_{z,w}\left[T_b(\mathrm{NW}^f_{m,d}(z)) - T_b(w)\right] \geq \delta$. 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 $\mathbb{E}\left[T_b(H_m) - T_b(H_0)\right] \geq \delta$. Pick $i \in_R [m]$ uniformly at random. Then we obtain $\mathbb{E}_i\left[T_b(H_i) - T_b(H_{i-1})\right] \geq \delta/m$.

We can exploit this advantage to predict the next bit of the PRG (due to Yao [Yao82]; a nice exposition can be found in [Vad12, 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 $\Pr[P^{T_b}(x) = f(x)] \geq \frac{1}{2} + \frac{\delta}{m}$, 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 $\Pr_{x \in_R \{0,1\}^\ell}\left[P^{T_b}(x) = f(x)\right] \geq \frac{1}{2} + \frac{\delta}{m}$.

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 $\sum_{j<i} 2^{|S_i \cap S_j|} + (m-i) + (d-\ell) + O(\log(md)) \leq \exp(\ell^2/d) \cdot m + d + O(\log(md))$. Thus we have $\mathrm{K}^T_{t,\,\delta/m}(f) \leq \rho(m,\ell,d) + O(\log(md))$.

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 $\Pr_{x \in_R \{0,1\}^\ell}\left[P^{T_b}(x) = f(x)\right] \geq \frac{1}{2} + \frac{\delta}{2m}$. 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 4.6 by incorporating a list-decodable error-correcting code, so that we obtain a certificate for $\mathrm{K}^T_t(x)$ instead of $\mathrm{K}^T_{t,\delta/2m}(f)$. 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 4.7 (List-decodable error-correcting code; see, e.g., [Sud97, STV01] and [Vad12, Problem 5.2])**.** *For any $n \in \mathbb{N}$ and $\epsilon > 0$, there exist functions $\mathrm{Enc}_{n,\epsilon}\colon \{0,1\}^n \to \{0,1\}^{2^\ell}$ and $\mathrm{Dec}_{n,\epsilon}\colon \{0,1\}^{2^\ell} \to (\{0,1\}^n)^L$ such that*

1. *$\ell = O(\log(n/\epsilon))$, $L = \mathsf{poly}(1/\epsilon)$,*
2. *for every $x \in \{0,1\}^n$ and $r \in \{0,1\}^{2^\ell}$, if $\mathrm{dist}(\mathrm{Enc}_{n,\epsilon}(x), r) \le \frac{1}{2} - \epsilon$ then we have $x \in \mathrm{Dec}_{n,\epsilon}(r)$, and*
3. *$\mathrm{Enc}_{n,\epsilon}$ and $\mathrm{Dec}_{n,\epsilon}$ can be computed 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 4.8. *$\mathrm{K}^T_{t'}(x) \le \mathrm{K}^T_{t,\,\epsilon}(\mathrm{Enc}_{n,\epsilon}(x)) + O(\log(n/\epsilon))$ for any string $x \in \{0,1\}^*$ of length $n \in \mathbb{N}$, any oracle T, and any $t' \ge t + \mathsf{poly}(n, 1/\epsilon)$. Moreover, given any x and any certificate for $\mathrm{K}^T_{t,\,\epsilon}(\mathrm{Enc}_{n,\epsilon}(x)) \preceq s$, one can find a certificate for $\mathrm{K}^T_{t'}(x) \preceq s + O(\log(n/\epsilon))$ in time $t + \mathsf{poly}(n, 1/\epsilon)$ with oracle access to T.*

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 $\mathrm{Dec}_{n,\epsilon}(U^T(d_0))$. By the definition, there exists some description d_0 of length $\mathrm{K}^T_{t,\,\epsilon}(\mathrm{Enc}_{n,\epsilon}(x))$ such that $U^T(d_0)$ 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 $\mathrm{K}^T_{t'}(x) \le |d_0| + O(\log(ni/\epsilon))$.

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

Now we combine Lemma 4.6 and the list-decodable error-correcting code.

Lemma 4.9. *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.*

Output: *A certificate for $\mathrm{K}^T_t(x) \preceq \rho(m, \ell, d) + O(\log(nmd/\delta))$ 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 4.6 with inputs $f, m, d/\ell, \delta^{-1}$ and oracle access to T. The algorithm outputs a certificate for $\mathrm{K}^T_{t_0,\,\delta/2m}(\mathrm{Enc}_{n,\epsilon}(x)) \preceq \rho(m, \ell, d) + O(\log(md))$, where $t_0 = \mathsf{poly}(m, d, 2^\ell)$. By Corollary 4.8, we may efficiently convert this certificate to a certificate for $\mathrm{K}^T_t(x) \preceq \rho(m, \ell, d) + O(\log(nmd/\epsilon))$, where $t \ge t_0 + \mathsf{poly}(n, 1/\epsilon)$. □

As a consequence of Lemma 4.9, for any $x \in \{0,1\}^*$ and parameters with $d \gg \ell^2$, we may obtain a certificate of length $\rho(m,\ell,d) \approx \exp(\ell^2/d) \cdot m + d \approx m + \ell^2 m/d + d$ given a statistical test for $\mathrm{NW}_{m,d}^{\mathrm{Enc}_{n,\epsilon}(x)}$. 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 4.10. *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*

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

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

Proof. 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 $\mathrm{NW}_{m,d}^f(z)$.

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

We now assume that an oracle T is a δ-dense subset of r-random strings $R_t[r]$. By Lemma 4.10, T is a distinguisher for $\mathrm{NW}_{m,d}^{\mathrm{Enc}_{n,\epsilon}(x)}$ if $\mathrm{K}_t(x) + d \lesssim r(m)$. Thus by Lemma 4.9 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 4.11. *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* $R_{t_1}[r]$ *for some* $t_1 \ge t + \mathsf{poly}(n, m, d, 1/\delta)$, *and that* $\mathrm{K}_t(x) + d + O(\log(nmd/\delta)) < r(m)$.

Output: *A certificate for* $\mathrm{K}_{t_2}^T(x) \preceq \rho(m,\ell,d) + O(\log(nmd/\delta))$ *for any* $t_2 \ge \mathsf{poly}(n, m, d, 1/\delta)$.

Proof. By Lemma 4.10, we have

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

for any $z \in \{0,1\}^d$ and any $t_1 \ge t + \mathsf{poly}(n, m, d, 1/\delta)$. Therefore, for any $z \in \{0,1\}^d$, the output $\mathrm{NW}_{m,d}^{\mathrm{Enc}_{n,\epsilon}(x)}(z)$ is not r-random with respect to K_{t_1}; in particular, $\mathrm{NW}_{m,d}^{\mathrm{Enc}_{n,\epsilon}(x)}(z) \notin T$. On the other hand, $\Pr_{w \in_R \{0,1\}^m}[w \in T] \ge \delta$ by the assumption. Thus T distinguishes $\mathrm{NW}_{m,d}^{\mathrm{Enc}_{n,\epsilon}(x)}$ from the uniform distribution with advantage at least δ. By running the algorithm of Lemma 4.9 with input $x, m, d/\ell, \delta^{-1}$, and oracle access to T, the algorithm outputs a certificate for

$$\mathrm{K}_{t_2}^T(x) \preceq \rho(m,\ell,d) + O(\log(nmd/\delta))$$

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

By Theorem 4.11, for $r(m) \approx m$, we can set $m \approx \mathrm{K}_t(x) + d$;[3] 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 $m - O(\sqrt{m}\log m) \le r(m)$.

Corollary 4.12. *Let $\delta^{-1} \in \mathbb{N}$ be any constant. Let $r\colon \mathbb{N} \to \mathbb{N}$ be any function such that $m - c\sqrt{m}\log m \le r(m)$, 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 $R_{t_1}[r]$ for some $t_1 \ge t + \mathsf{poly}(n)$.*

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 $t_2 \ge \mathsf{poly}(n)$.*

Proof. Without loss of generality, we may assume that $O(\log n) \le \mathrm{K}_t(x) \le n + O(1)$. 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 $\mathrm{K}_t(x) \ge n + O(1)$, then we may just output a trivial description for x of length $n + O(1)$. In what follows, we assume $O(\log n) \le \mathrm{K}_t(x) \le n + O(1)$.

Here is the algorithm: For every $m \in \{1, \dots, n + O(1)\}$ and every $d/\ell \in \{1, \dots, n + O(1)\}$, run the algorithm of Theorem 4.11 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 4.11 outputs a short description. Let $\ell := \log|\mathrm{Enc}_{n,\delta/2m}(x)| = O(\log n)$. We analyze the two cases depending on whether $\ell \le \sqrt{\mathrm{K}_t(x)}$ or not. Consider the case when $\ell \le \sqrt{\mathrm{K}_t(x)}$. 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 $d \le \mathrm{K}_t(x)$ and hence $m \le 4\mathrm{K}_t(x)$. Thus,

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

which means the hypothesis of Theorem 4.11 is satisfied; therefore, with high probability, the algorithm outputs a description d_0 for x such that $|d_0| \le \exp(\ell^2/d) \cdot m + d + O(\log n)$. Since $\ell^2/d \le 1$, the length of the description is

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

[3] The extra term d can be removed if the candidate pseudorandom generator is defined as $z \mapsto (z, \mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}(z))$. However, it does not improve the quality of the approximation significantly.

Next, consider the case when $\ell > \sqrt{\mathrm{K}_t(x)}$. In this case, let $d/\ell := \ell$ and $m := 4d$. Then we have $r(m) \ge m - c\sqrt{m}\log m \ge 3d > \mathrm{K}_t(x) + d + O(\log n)$, which confirms the hypothesis of Theorem 4.11. Thus the algorithm outputs a description for x of length $\exp(1) \cdot m + d + O(\log n) = O(\ell^2)$. $\square$

4.2 Accepting a Dense Subset of Random Strings in Heuristica

Now we justify the hypothesis used in the previous subsection. 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 define some specific problem $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}})$ in DistNP. We consider the following distribution on a pair $(x, 1^t)$ of a binary string and a unary string.

Definition 4.13 (Uniform distribution with auxiliary unary input)**.** *Define a family of distributions $\mathcal{D}^{\mathrm{KT}} := \{\mathcal{D}^{\mathrm{KT}}_m\}_{m\in\mathbb{N}}$, where, for each $m \in \mathbb{N}$, $\mathcal{D}^{\mathrm{KT}}_m$ is defined as the output distribution of the following algorithm: Pick $n \in_R [m]$ and $x \in_R \{0,1\}^n$ randomly. Output $(x, 1^{m-n})$.*

From this definition, it is obvious that $\mathcal{D}^{\mathrm{KT}}$ 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 $\mathcal{D}^{\mathrm{KT}}_m$. Given an input $(x, 1^t)$ sampled from $\mathcal{D}^{\mathrm{KT}}$, we consider a problem of deciding whether x is r-nonrandom string with respect to K_t.

Definition 4.14. *For a function $r\colon \mathbb{N} \to \mathbb{N}$, define*

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

DistNP is the class of distributional problems $(L, \mathcal{D})$ such that $L \in \mathsf{NP}$ and $\mathcal{D}$ is efficiently samplable. It is easy to see that $\mathrm{MINKT}[r] \in \mathsf{NP}$, and thus:

Fact 4.15. $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}}) \in \mathsf{DistNP}$ *if $r\colon \mathbb{N} \to \mathbb{N}$ is efficiently computable.*

Next, we formally define the notion of errorless algorithm solving a distributional problem.

Definition 4.16 (Errorless Heuristic Algorithm; cf. [BT06a])**.** *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 $\bot$ for every $x \in \{0,1\}^*$, and*
- $\Pr_{x\sim\mathcal{D}_m}[A(x) = \bot] \le \delta(m)$ *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 $\mathsf{AvgP} := \bigcap_{c\in\mathbb{N}} \mathsf{Avg}_{m^{-c}}\mathsf{P}$.

We claim that if $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}})$ 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 4.17. *Let $r\colon \mathbb{N} \to \mathbb{N}$ be any function such that $r(n) < n$ for all large $n \in \mathbb{N}$. If $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}}) \in \mathsf{Avg}_\delta\mathsf{P}$ for $\delta(m) := 1/6m$, then there exists a language $T \in \mathsf{P}$ such that $T^{=n}_t$ 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 an errorless heuristic deterministic polynomial-time algorithm for $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}})$. 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 4.18. *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

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

where in the last inequality we used the fact that, conditioned on the event $|x| = n$, the distribution $\mathcal{D}^{\mathrm{KT}}_m$ 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 4.18.

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 4.12; 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 4.19. *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 $\mathrm{K}_{t_3}(x) \preceq |d_0| + O(\log\log t_1)$. □

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. [NW94, IW97, KvM02]); however, for our purpose, there is a much simpler construction of a pseudorandom generator based on Lemma 4.9. (Our construction is similar to the Sudan-Trevisan-Vadhan pseudorandom generator [STV01].)

Lemma 4.20. *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 4.9, 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. □

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

Theorem 4.21. *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 $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}}) \in \mathsf{Avg}_{1/6m}\mathsf{P}$. 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 4.17, 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 4.12 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 4.19, 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 3.9, 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 4.20. 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 4.20. 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 4.20, 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 4.22. *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. $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}}) \in \mathsf{Avg}_{1/6m}\mathsf{P}$ *for some $r\colon \mathbb{N} \to \mathbb{N}$ such that $n - O(\sqrt{n}\log n) \leq r(n) < n$ 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 $\sigma(n,s) = s + O((\log n)\sqrt{s} + (\log n)^2)$ and some polynomial $\tau(n,t)$.*
4. $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in$ Promise-ZPP *for some $\sigma(n,s) = s + O((\log n)\sqrt{s} + (\log n)^2)$ and some polynomial $\tau(n,t)$.*

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

($2 \Rightarrow 3$) This is exactly equivalent to Theorem 4.21.

($3 \Rightarrow 4$) This follows from Fact 3.8.

($4 \Rightarrow 2$ 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 $\sigma(n,s) \leq s +$

$c \cdot \big((\log n)\sqrt{s} + (\log n)^2\big)$ for all large $n, s \in \mathbb{N}$. We claim that $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}}) \in \mathsf{Avg}_\delta\mathsf{P}$ 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 $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$. Using the algorithm, we define an errorless heuristic algorithm A solving $\mathrm{MINKT}[r]$ as follows: On input $(x, 1^t)$, if the input is short, i.e., $|x| = O(1)$, then check if $(x, 1^t) \in \mathrm{MINKT}[r]$ 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 $(x, 1^t) \in \mathrm{MINKT}[r]$. Since $\mathrm{K}_t(x) < r(|x|) = s$, $(x, 1^t, 1^s)$ is a YES instance of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$; 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 $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$. 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

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

□

Corollary 4.23. *If* $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ *then* $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in \mathrm{Promise}\text{-}\mathrm{P}$ *for some* $\sigma(n, s) = s + O\big((\log n)\sqrt{s} + (\log n)^2\big)$ *and some polynomial* $\tau(n, t)$*.*

Proof. Buhrman, Fortnow and Pavan [BFP05] showed that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ implies the existence of a pseudorandom generator, and in particular, Promise-BPP = Promise-P. Corollary 4.23 immediately follows by combining their result with Corollary 4.22. □

We leave it as an open question to derandomize the search to average-case reduction of Theorem 4.21 without relying on the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$. In the next theorem, we show that a *decision* to average-case reduction can be derandomized at the cost of the quality of the approximation; specifically, the problem of approximating $\mathrm{K}_t(x)$ within a factor of n^γ for an arbitrary constant $\gamma > 0$ is reducible to $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}})$ via a deterministic reduction.

Theorem 4.24. *Let* $r(n) := n/2$*. Let* $\gamma > 0$ *be an arbitrary constant, and define* $\sigma(n, s) := n^\gamma s$*. If* $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}}) \in \mathsf{Avg}_{1/6m}\mathsf{P}$ *then* $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in \mathrm{Promise}\text{-}\mathrm{P}$ *for some polynomial* τ*.*

Proof. The idea is to exhaustively search all the seeds z of $\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}(z)$ and compute the acceptance probability $p := \Pr_{z \in_R \{0,1\}^d}\left[T'(\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}(z)) = 1\right]$ deterministically, where $T' \in \mathsf{P}$ is a dense subset of random strings obtained from Lemma 4.17. Then by Lemma 4.10, $p = 0$ if x is not random; conversely, by the security proof of the pseudorandom generator (Lemma 4.9), if $p \approx 0$, then x can be described by a small program, as T' is a statistical test for the pseudorandom generator $\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}$. Therefore, p characterizes the Kolmogorov complexity of x. Details follow.

We describe a polynomial-time algorithm for computing $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$: Let $(x, 1^t, 1^s)$ be any instance of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$. Define $\delta := 1/6$ and $m := 4s$ (so that $2s \le r(m)$). Let c be an arbitrary constant, and take the parameter $d/\ell := c$. Recall that 2^ℓ is the length of $\mathrm{Enc}_{n,\epsilon}(x)$, where $\epsilon = \delta/2m$. In particular, since the length of a seed z is $d = c\ell = O(\log n)$, one can enumerate all the seeds $z \in \{0,1\}^d$ in polynomial time. For simplicity of the notation, let $G(z)$ denote $\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}(z)$. We will choose a parameter $t' \in \mathbb{N}$ later, and define $T' := T^{=m}_{t'}$, where $T \in \mathsf{P}$ is the set of Lemma 4.17. The algorithm computes $p := \Pr_{z \in_R \{0,1\}^d}[T'(G(z)) = 1]$ by enumerating all the seeds z, and accepts if and only if $p = 0$.

The correctness of this algorithm immediately follows from the next claim.

Claim 4.25.

1. *If* $\mathrm{K}_t(x) \le s$ *then* $p = 0$.
2. *If* $\mathrm{K}_{\mathsf{poly}(n,t')}(x) > n^{O(1/c)} \cdot s$ *then* $p \ge \frac{1}{6}$.

To prove Item 1, assume that $\mathrm{K}_t(x) \le s$. By Lemma 4.10, for every seed z, we have

$$\mathrm{K}_{t'}(G(z)) \le \mathrm{K}_t(x) + d + O(\log n) < 2s,$$

where in the last inequality we used the fact that we may assume without loss of generality that $O(\log n) \le \mathrm{K}_t(x) \le n + O(1)$. In particular, we have $\mathrm{K}_{t'}(G(z)) < 2s \le r(m)$ for some $t' = t + \mathsf{poly}(n)$. Since $T^{=m}_{t'}$ is a subset of $R_{t'}[r]$, we have $G(z) \notin T^{=m}_t$ for every seed z; thus $p = 0$. This completes the proof of Item 1.

Next, we prove the contrapositive of Item 2. Suppose that $p < \frac{1}{6}$. Since $T^{=m}_{t'}$ is $\frac{1}{3}$-dense, $T^{=m}_{t'}$ distinguishes $\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}$ from the uniform distribution with advantage $\frac{1}{3} - \frac{1}{6} = \frac{1}{6}$. Applying Lemma 4.9, we obtain that $\mathrm{K}^{T_{t'}}_{\mathsf{poly}(n)}(x) \le \exp(\ell^2/d) \cdot m + O(\log n) \le \exp(O(\log n)/c) \cdot 4s \le n^{O(1/c)} \cdot s$. Since $T \in \mathsf{P}$, it follows from Lemma 4.19 and Lemma 3.9 that $\mathrm{K}_{\mathsf{poly}(n,t')}(x) \le n^{O(1/c)} \cdot s + O(\log\log t') \le n^{O(1/c)} \cdot s$. This completes the proof of Item 2.

Since c is an arbitrary constant, by taking c large enough, we obtain an approximation factor of n^γ for an arbitrary constant $\gamma > 0$. □

5 Hardness of GapMINKT

In this section, we present evidence against $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in \mathsf{coNP}$. We start with the definition of hitting set generator. In contrast to the fact that a secure pseudorandom generator can be used to derandomize two-sided-error randomized algorithms, a secure hitting set generator can be used to derandomize one-sided-error randomized algorithms.

Definition 5.1 (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* $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ *is said to* γ-avoid G *if for every* $n \in \mathbb{N}$, $G_n(z) \in \Pi_{\mathrm{NO}}$ *for any* $z \in \{0,1\}^{s(n)}$, *and* $\Pr_{w\in_R\{0,1\}^{t(n)}}[w \in \Pi_{\mathrm{YES}}] \ge \gamma(n)$. *$G$ is called a* hitting set generator γ-secure against a complexity class $\mathfrak{C}$ *if there is no promise problem* $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}}) \in \mathfrak{C}$ *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 $G := \{G_n : \{0,1\}^{s(n)} \to \{0,1\}^{t(n)}\}_{n\in\mathbb{N}}$ is *efficiently computable* if there exists a polynomial-time algorithm that, given as input $z \in \{0,1\}^{s(n)}$ and $n \in \mathbb{N}$, computes $G_n(z)$ in time $\mathsf{poly}(t(n))$.

It is easy to see that there is no efficiently computable hitting set generator γ-secure against coNP for any reasonable choice of γ, by guessing a seed. 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 satisfying the promise of GapMINKT. For simplicity, we focus on the case of $t(n) = n$.

Theorem 5.2. *Let $\sigma, \tau : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be any functions such that $\sigma(n, s) \ge s$ for any $n, s \in \mathbb{N}$. Let $G = \{G_n : \{0,1\}^{s(n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ be any family of functions computable in time $\mathsf{poly}(n)$, where $s : \mathbb{N} \to \mathbb{N}$ is an efficiently computable function. Let $\gamma : \mathbb{N} \to [0,1]$ be any function such that $\sigma(n, s(n) + O(\log n)) \le n - 1 + \log(1 - \gamma(n))$ for all large $n \in \mathbb{N}$. 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 $A \subseteq \{0,1\}^*$ satisfying the promise of* $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$.

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 a 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 4.21, $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ is capable of avoiding any efficiently computable hitting set generator $G = \{G_n : \{0,1\}^{s(n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ such that $s(n) \le n - c\sqrt{n}\log n$ for some large constant $c > 0$.

Corollary 5.3. *Let $c > 0$ be an arbitrary constant. Let $\sigma, \tau : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be any functions such that $s \le \sigma(n, s) \le s + c\big((\log n)\sqrt{s} + (\log n)^2\big)$ for all large $n, s \in \mathbb{N}$. If* $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in \text{Promise-}\mathsf{P}$ *then there is no efficiently computable hitting set generator $G = \{G_n : \{0,1\}^{s'(n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ γ-secure against P, where s' and γ are arbitrary functions satisfying $s'(n) \le n - 2c\sqrt{n}\log n + \log(1 - \gamma(n))$ for all large $n \in \mathbb{N}$.*

Proof. We verify that the assumption of Theorem 5.2 is satisfied. Indeed, $s'(n) + O(\log n) \le n - 2c\sqrt{n}\log n + O(\log n) \le n$ for all large n, and therefore,

$$\begin{aligned}
&\sigma(n, s'(n) + O(\log n)) \\
&\le s'(n) + O(\log n) + c\big((\log n)\sqrt{n} + (\log n)^2\big) \\
&\le n - 1 + \log(1 - \gamma(n)).
\end{aligned}$$

□

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 [RR97], can be cast as algorithms avoiding a specific hitting set generator. The hitting set generator is defined as follows.

Definition 5.4 (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 *$G^{\mathsf{int},s}_\ell$ of 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 5.5 (Γ-natural property). *A promise problem Π is called a* natural property useful against $\mathsf{SIZE}(s(\ell))$ with largeness γ *if Π γ-avoids the circuit interpreter $G^{\mathsf{int},s}$ of parameter s. If, in addition, $\Pi \in \mathfrak{C}$ for a complexity class $\mathfrak{C}$, then Π is called a* $\mathfrak{C}$-natural property*.*

Rudich [Rud97] conjectured that there is no $\mathsf{NP}/\mathsf{poly}$-natural property useful against P/poly. In our terminology, his conjecture implies that $G^{\mathsf{int},s}$ is a hitting set generator secure against $\mathsf{NP}/\mathsf{poly}$ for any $s(\ell) = \ell^{\omega(1)}$. Thus his conjecture implies $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \notin \mathsf{coNP}/\mathsf{poly}$ for a wide range of parameters σ.

Corollary 5.6. *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 $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in \mathsf{coNP}/\mathsf{poly}$, then there is some $\mathsf{NP}/\mathsf{poly}$-natural property useful against P/poly with largeness $\frac{1}{2}$.*

Proof. We apply Theorem 5.2 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 [RR97]: 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 $G^{\mathsf{int},s'}_\ell$ 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. □

5.2 Random 3SAT as a Hitting Set Generator

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 specific hitting set generator $G^{\mathrm{R3SAT}} = \{G^{\mathrm{R3SAT}}_N \colon \{0,1\}^{N-\Omega(N/\log N)} \to \{0,1\}^N\}_{N\in\mathbb{N}}$. Random 3SAT is defined as follows.

Definition 5.7 (Random 3SAT)**.** *Let $m\colon \mathbb{N} \to \mathbb{N}$ be an efficiently computable function. Let $\mathcal{D}^{\mathrm{R3SAT}} = \{\mathcal{D}_n^{\mathrm{R3SAT}}\}_{n\in\mathbb{N}}$ be a family of the following distributions $\mathcal{D}_n^{\mathrm{R3SAT}}$ of 3CNF formulas: Let n be the number of variables, and $m = m(n)$ be the number of clauses. The distribution $\mathcal{D}_n^{\mathrm{R3SAT}}$ samples a random n-variable m-clause 3CNF formula φ by choosing each clause independently and uniformly at random from all the possible $2^3\binom{n}{3}$ width-3 clauses on n variables. We say that a promise problem $\Pi = (\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ solves Random 3SAT with probability $\gamma(n)$ if every satisfiable formula is in Π_{YES} and $\Pr_{\varphi\sim\mathcal{D}_n^{\mathrm{R3SAT}}}[\varphi \in \Pi_{\mathrm{NO}}] \geq \gamma(n)$ for every $n \in \mathbb{N}$.*

The proof idea of Random 3SAT-hardness of MKTP [HS17] is that every satisfiable formula can be compressed. We describe it in a slightly different way.

Lemma 5.8 ([HS17])**.** *There exists an efficiently computable family of functions*

$$G^{\mathrm{R3SAT}} = \{G_n^{\mathrm{R3SAT}} : \{0,1\}^{n+\lceil m(n)\log(7\binom{n}{3})\rceil} \to \{0,1\}^{\lceil m(n)\log(8\binom{n}{3})\rceil}\}_{n\in\mathbb{N}}$$

such that the image of G_n^{R3SAT} contains all the satisfiable n-variable $m(n)$-clause 3CNF formula. (Here we regard the output of G_n^{R3SAT} as an encoding of a 3CNF formula.)

Proof. Fix any $n \in \mathbb{N}$, and fix any satisfiable n-variable $m(n)$-clause 3CNF formula φ. Let $a \in \{0,1\}^n$ be some satisfying assignment of φ. We claim that φ can be described by the assignment a and some auxiliary information $d \in \{0,1\}^{\lceil m(n)\log(7\binom{n}{3})\rceil}$. ($G_n^{\mathrm{R3SAT}}(a,d)$ is defined as the output of a description procedure below.)

Indeed, given a satisfying assignment a of φ, there are $(2^3-1)\binom{n}{3}$ possible clauses that are satisfied by a. More specifically, fix any 3 variables of a clause; then there are 7 ways to negate these variables so that the resulting clause is satisfied by a. (For example, if the assignment a is $\{x_1 \mapsto 0, x_2 \mapsto 1, x_3 \mapsto 0\}$, then $x_1 \vee \neg x_2 \vee x_3$ is the unique clause that is not satisfied by a.) Therefore, each clause of φ can be described with $\log(7\binom{n}{3})$ bits, and hence φ can be described with some auxiliary information d of length $m(n)\log(7\binom{n}{3})$. □

We observe that solving Random 3SAT is essentially equivalent to avoiding G^{R3SAT}.

Proposition 5.9 (Random 3SAT and G^{R3SAT})**.** *Let $\Pi = (\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ be any promise problem and $\overline{\Pi} = (\Pi_{\mathrm{NO}}, \Pi_{\mathrm{YES}})$ be its complement. If Π γ-avoids G^{R3SAT}, then $\overline{\Pi}$ solves Random 3SAT with probability $1 - 2(1-\gamma(n))$.*

Proof. Let φ be any satisfiable formula. By Lemma 5.8, there exists a description (d,a) such that $G_n^{\mathrm{R3SAT}}(d,a) = \varphi$; since Π avoids G^{R3SAT}, we obtain $\varphi \in \Pi_{\mathrm{NO}}$. Therefore, the promise problem $\overline{\Pi}$ accepts every satisfiable formula.

Next, we claim that $\overline{\Pi}$ rejects most formulas. Let $N := m(n)\log(8\binom{n}{3})$. Since Π γ-avoids G^{R3SAT}, $w \in \Pi_{\mathrm{YES}}$ holds with probability at least $\gamma(n)$ over the choice of a string $w \in_R \{0,1\}^{\lceil N\rceil}$. Let $D \subseteq \{0,1\}^{\lceil N\rceil}$ be the set of all the encodings of n-variable $m(n)$-clause 3CNF formulas. In particular, $|D| = 2^N$ and hence $\Pr_{w\in_R\{0,1\}^{\lceil N\rceil}}[w \in D] \geq 2^{N-\lceil N\rceil} \geq \frac{1}{2}$. Now, since

$$1-\gamma(n) \geq \Pr_{w\in_R\{0,1\}^{\lceil N\rceil}}[w \notin \Pi_{\mathrm{YES}}] \geq \Pr_{w\in_R\{0,1\}^{\lceil N\rceil}}[w \in D] \cdot \Pr_{w\in_R D}[w \notin \Pi_{\mathrm{YES}}],$$

we obtain $\Pr_{w\in_R D}[w \notin \Pi_{\mathrm{YES}}] \leq 2(1-\gamma(n))$. □

Corollary 5.10 (Random 3SAT-hardness of GapMINKT)**.** *Let Δ be any constant such that $\Delta > 1/\log(8/7)$. Let $m(n) = \Delta n$ be the number of clauses. Then,* $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ *is Random 3SAT-hard, where σ and τ are any functions as in Corollary 5.3. In particular, if* $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in \mathsf{coNP}$*, then there exists a* coNP *algorithm solving Random 3SAT of $m(n)$ clauses with probability $1 - 2^{-\Omega(n)}$, where n denotes the number of variables.*

Proof. Let N denote the output length of G_n^{R3SAT}, that is, $N := \lceil m(n)\log(8\binom{n}{3})\rceil$. The seed length of G_n^{R3SAT} is $s(N) := n + \lceil m(n)\log(7\binom{n}{3})\rceil \le N - (\Delta\log(8/7) - 1)n + 1 \le N - \Omega(n) \le N - \Omega(N/\log N)$. Applying Corollary 5.3 to G_n^{R3SAT} and using Proposition 5.9, we obtain Random 3SAT-hardness of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$. □

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 6.1 (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 6.2 (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 6.3 (Decision reduces to search for MCSP)**.** *If there exists a randomized polynomial-time 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 3.8. 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 6.4 (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 6.5 (Randomized Errorless Heuristics; cf. [BT06a]))**.** *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*

- $\Pr_A\big[A(x) \notin \{L(x), \bot\}\big] \le \frac{1}{8}$ *for every* $x \in \{0,1\}^*$, *and*
- $\Pr_{x \sim U_n}\big[\Pr_A[A(x) = \bot] > \frac{1}{8}\big] \le \delta(n)$ *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 [HS17], we show that an errorless heuristic algorithm for MCSP$[s]$ is essentially equivalent to BPP-natural properties useful against $\mathsf{SIZE}(s(n))$.

Lemma 6.6. *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 $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ be a BPP-natural property, and M be a BPP algorithm solving $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ (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 $\bot$ 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 $\Pr_A[A(f) = \bot] = \Pr_M[M(f) = 0] > \frac{1}{8}$, which implies that $f \notin \Pi_{\mathrm{YES}}$. 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 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 $f \in \Pi_{\mathrm{NO}}$ 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 $A(f) = \bot$; otherwise $M(f) := 1$. We claim that M accepts some natural property $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ 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 $\bot$ 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 6.7. *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 6.6.

(2 $\Rightarrow$ 3) Obvious.

(6 $\Rightarrow$ 1) This follows from Fact 6.3.

(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 a 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 [CIKK16]. We prove this in the next Theorem 6.8. □

Theorem 6.8. *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 [CIKK16], which established a generic reduction from learning an ϵ-close function to natural properties.

Lemma 6.9 (Carmosino, Impagliazzo, Kabanets, and Kolokolova [CIKK16])**.** *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,*

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

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 6.8. 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 $|C'| \leq |C| + \epsilon \cdot 2^n \cdot n + O(1)$. 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 6.9 can be converted to a circuit C' computing f exactly such that $|C'| \leq \mathsf{poly}(n, 1/\epsilon, 2^\ell, |D|) + \epsilon \cdot 2^n \cdot n$.

We now construct a statistical test for $G^f_{\ell,n,\epsilon}$ using a BPP-natural property and Adleman's trick [Adl78] (for proving $\mathsf{BPP} \subseteq \mathsf{P/poly}$). Let $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$ be a BPP-natural property and M be a BPP algorithm solving $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$. 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 $w \in \Pi_{\text{YES}}$, for every $w \in (\Pi_{\text{YES}} \cup \Pi_{\text{NO}})^{=L}$. 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 $\mathsf{size}(G^f_{\ell,n,\epsilon}(z)) \leq u(\ell)$ for every z. Indeed, by the usefulness of natural properties, we have $G^f_{\ell,n,\epsilon}(z) \in \Pi_{\text{NO}}$; thus $D_\ell(G^f_{\ell,n,\epsilon}(z)) = 0$. On the other hand, by the largeness of natural properties, we have $|(\Pi_{\text{YES}})^{=L}| \geq \delta_0 2^L$, and thus $\Pr_{w \in_R \{0,1\}^L}[D_\ell(w) = 1] \geq \delta_0$. 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 6.9 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 $G^f_{\ell,n,\epsilon}(z) \leq (ns/\epsilon)^c$ and $|C'| \leq (n2^\ell/\epsilon)^c + \epsilon 2^n n$ 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 $(ns/\epsilon)^c \leq u(\ell) = 2^{\epsilon_0 \ell}$; 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 $|C'| \leq (n/\epsilon)^c (n 2^{\epsilon_1 n})^{c^2/\epsilon_0} + 2^{(1-\epsilon_1)n} ns \leq n^{O(1)} 2^{(1-\epsilon_1)n} s \leq 2^{(1-\epsilon_1/2)n} s$ for all large $n \in \mathbb{N}$. $\square$

Acknowledgment

We thank Rahul Santhanam for helpful discussion, and thank Eric Allender, Oded Goldreich, and the anonymous reviewers for helpful comments on manuscripts.

References

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

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

[Adl78] Leonard M. Adleman. Two Theorems on Random Polynomial Time. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 75–83, 1978.

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

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

[AH17] 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.

[AHK17] Eric Allender, Dhiraj Holden, and Valentine Kabanets. The Minimum Oracle Circuit Size Problem. *Computational Complexity*, 26(2):469–496, 2017.

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

[Ajt96] 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.

[AKRR11] 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.

[AR05] Dorit Aharonov and Oded Regev. Lattice problems in $\mathrm{NP} \cap \mathrm{coNP}$. *J. ACM*, 52(5):749–765, 2005.

[AW09] Scott Aaronson and Avi Wigderson. Algebrization: A New Barrier in Complexity Theory. *TOCT*, 1(1):2:1–2:54, 2009.

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

[BFNW93] 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.

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

[BGS75] Theodore P. Baker, John Gill, and Robert Solovay. Relativizations of the P =? NP Question. *SIAM J. Comput.*, 4(4):431–442, 1975.

[BGSV16] Peter Bürgisser, Oded Goldreich, Madhu Sudan, and Salil Vadhan. Complexity Theory. *Oberwolfach Reports*, 12(4):3049–3099, 2016.

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

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

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

[CIKK16] 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.

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

[Fei02] Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 534–543, 2002.

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

[FKO06] 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.

[FO07] Uriel Feige and Eran Ofek. Easily refutable subformulas of large random 3CNF formulas. *Theory of Computing*, 3(1):25–43, 2007.

[GG00] Oded Goldreich and Shafi Goldwasser. On the Limits of Nonapproximability of Lattice Problems. *J. Comput. Syst. Sci.*, 60(3):540–563, 2000.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GST07] 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.

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

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

[HP15] 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.

[HR12] 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.

[HS17] 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.

[HW16] 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.

[HW19] Shuichi Hirahara and Osamu Watanabe. On Nonadaptive Reductions to the Set of Random Strings and Its Dense Subsets. Manuscript, 2019.

[IKV18] 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.

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

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

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

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

[IW01] Russell Impagliazzo and Avi Wigderson. Randomness vs Time: Derandomization under a Uniform Assumption. *J. Comput. Syst. Sci.*, 63(4):672–688, 2001.

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

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

[KvM02] 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.

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

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

[MR07] Daniele Micciancio and Oded Regev. Worst-Case to Average-Case Reductions Based on Gaussian Measures. *SIAM J. Comput.*, 37(1):267–302, 2007.

[MW17] Cody D. Murray and R. Ryan Williams. On the (Non) NP-Hardness of Computing Circuit Complexity. *Theory of Computing*, 13(1):1–22, 2017.

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

[OS17] 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.

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

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

[RRV02] 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.

[Rud97] 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.

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

[Sud97] Madhu Sudan. Decoding of Reed Solomon Codes beyond the Error-Correction Bound. *J. Complexity*, 13(1):180–193, 1997.

[Tra84] 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.

[Tre01] Luca Trevisan. Extractors and pseudorandom generators. *J. ACM*, 48(4):860–879, 2001.

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

[Vad06] Salil P. Vadhan. An Unconditional Study of Computational Zero Knowledge. *SIAM J. Comput.*, 36(4):1160–1214, 2006.

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

[Yao82] 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.

[Zim05] Marius Zimand. A List-Decodable Code with Local Encoding and Decoding. In *Proceedings of the 6th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2005)*, pages 232–237, 2005.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il