THEORY OF COMPUTING, Volume 19 (4), 2023, pp. 1–61
www.theoryofcomputing.org

SPECIAL ISSUE: CCC 2020

Non-Disjoint Promise Problems from Meta-Computational View of Pseudorandom Generator Constructions

Shuichi Hirahara*

Received August 18, 2020; Revised September 26, 2022; Published October 14, 2023

Abstract. The standard notion of promise problem is a pair of *disjoint* sets of instances, each of which is regarded as YES and NO instances, respectively, and the task of solving a promise problem is to distinguish these two sets of instances. In this paper, we introduce a set of new promise problems which are conjectured to be *non-disjoint*, and prove that hardness of these "non-disjoint" promise problems gives rise to the existence of hitting set generators (and vice versa). We do this by presenting a general principle which converts any black-box construction of a pseudorandom generator into the existence of a hitting set generator whose security is based on hardness of some "non-disjoint" promise problem (via a non-black-box security reduction).

Applying the principle to cryptographic pseudorandom generators, we introduce the Gap($\mathrm{K}^{\mathsf{SAT}}$ vs K) problem, which asks to distinguish strings whose time-bounded

A conference version of this paper appeared in the Proceedings of the 35th Computational Complexity Conference, 2020 [34].

*Supported by ACT-I, JST and JSPS KAKENHI Grant Number 18H04090.

ACM Classification: F.1.3

AMS Classification: 68Q15

Key words and phrases: pseudorandom generator, hitting set generator, meta-complexity, Levin's Kt complexity, minimum circuit size problem

 DOI: 10.4086/toc.2023.v019.a004

SAT-oracle Kolmogorov complexity is small from strings whose time-bounded Kolmogorov complexity (without SAT oracle) is large. We show that if this problem is NP-hard, the worst-case and average-case complexities of PH are equivalent. This generalizes the non-black-box worst-case to average-case reductions of Hirahara (FOCS'18) and improve the approximation error from $\widetilde{O}(\sqrt{n})$ to $O(\log n)$.

Applying the principle to complexity-theoretic pseudorandom generators, we introduce a family of Meta-computational Circuit Lower-bound Problems (MCLPs), which are problems of distinguishing the truth tables of explicit functions from hard functions. Our results generalize the hardness versus randomness framework and identify problems whose circuit lower bounds characterize the existence of hitting set generators.

We also establish the equivalence between the existence of a non-trivial derandomization algorithm for uniform algorithms and a uniform lower bound for a problem of approximating Levin's Kt-complexity.

1 Introduction

A *promise problem*, introduced by Even, Selman, and Yacobi [19], is a pair of disjoint sets $(\Pi_{\text{Yes}}, \Pi_{\text{No}})$ that are regarded as the sets of YES and NO instances, respectively. The problem is regarded as a problem whose instances are "promised" to come from $\Pi_{\text{Yes}} \cup \Pi_{\text{No}}$. Specifically, an algorithm A is said to *solve* a promise problem $(\Pi_{\text{Yes}}, \Pi_{\text{No}})$ if A accepts any instance $x \in \Pi_{\text{Yes}}$ and rejects any instance $x \in \Pi_{\text{No}}$; the behavior of A on any "unpromised" instance $x \notin \Pi_{\text{Yes}} \cup \Pi_{\text{No}}$ can be arbitrary. The notion of promise problem is crucial for formalizing several important concepts and theorems in complexity theory. A canonical example is the unique satisfiability problem (1SAT, UNSAT), where 1SAT is the set of formulas that has a unique satisfying assignment, and UNSAT is the set of unsatisfiable formulas. The promise problem is a standard Promise-UP-complete problem, and the celebrated theorem of Valiant and Vazirani [75] states that it is in fact NP-hard under randomized reductions. The reader is referred to the survey of Goldreich [24] for more background on promise problems.

Usually, it is required that Π_{Yes} and Π_{No} are disjoint, i. e., $\Pi_{\text{Yes}} \cap \Pi_{\text{No}} = \emptyset$. The reason is that if there exists an instance $x \in \Pi_{\text{Yes}} \cap \Pi_{\text{No}}$, then no algorithm can solve the promise problem $(\Pi_{\text{Yes}}, \Pi_{\text{No}})$. Indeed, if there were an algorithm A that solves $(\Pi_{\text{Yes}}, \Pi_{\text{No}})$, then A must accept x and simultaneously reject x, which is impossible. For this reason, every definition of promise problems considered before is, to the best of our knowledge, always disjoint.

In this paper, we introduce a set of new promise problems which are *conjectured to be non-disjoint*. We will demonstrate that these "non-disjoint" promise problems are worth investigating, by showing that hardness results for our promise problems have important consequences in complexity theory. The fact that the promise problems are conjectured to be non-disjoint means that solving promise problems are conjectured to be *impossible*, no matter how long an algorithm is allowed to run. Nevertheless, under several (implausible) assumptions, we show that "non-disjoint" promise problems can be *efficiently solved*; in other words, the promise problems are not only disjoint but also can be shown to be disjoint in a "constructive" way.

Taking the contrapositive, we prove that mild hardness results for computing "non-disjoint" promise problems, which is conjectured to be *impossible*, are sufficient to resolve important open questions of complexity theory.

To be more specific, we consider open questions of whether there exists an explicit hitting set generator. A *hitting set generator* (HSG) $G = \{G_n \colon \{0,1\}^{s(n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ secure against a class $\mathfrak{C}$ is a family of functions G_n such that any algorithm $A \in \mathfrak{C}$ that accepts at least a half of the n-bit strings must accept a string $G_n(z)$ for some seed $z \in \{0,1\}^{s(n)}$, for all large $n \in \mathbb{N}$. The existence of a secure hitting set generator makes it possible to derandomize any one-sided-error $\mathfrak{C}$-randomized algorithm, by simply trying all possible $s(n)$-bit seeds z and using $G_n(z)$ as a source of randomness. A stronger notion called a *pseudorandom generator* (PRG) enables us to derandomize two-sided-error randomized algorithms.

1.1 Meta-computational view of PRG constructions

A standard approach for constructing pseudorandom generators is to use the *hardness versus randomness* framework developed in, e. g., [77, 7, 56, 6, 41, 44, 68, 48]. One of the landmark results of Impagliazzo and Wigderson [44] states that if there exists a function in $\mathsf{E} = \mathsf{DTIME}(2^{O(n)})$ that is not computable by a circuit of size $2^{\alpha n}$ for some constant $\alpha > 0$, then there exists a logarithmic-seed-length pseudorandom generator secure against linear-size circuits (and, in particular, $\mathsf{P} = \mathsf{BPP}$ follows). In general, such a result is proved by using a *black-box pseudorandom generator construction* $G^{(\cdot)}$ that converts any hard function $f \notin \mathsf{SIZE}(2^{o(n)})$ to a pseudorandom generator $G^f \colon \{0,1\}^{O(n)} \to \{0,1\}^{2^{\alpha n}}$ secure against circuits of size $2^{\alpha n}$, where $\alpha > 0$ is some constant.

The underlying theme of this paper is to view black-box PRG constructions from a *meta-computational* perspective. Usually, f is regarded as a *fixed* hard function such as $f \notin \mathsf{SIZE}(2^{o(n)})$. Instead, here we regard f as an *input* to some "non-disjoint" promise problem, and regard a black-box PRG construction $G^{(\cdot)}$ as a reduction that proves the security of some (universal) hitting set generator based on the hardness of the "non-disjoint" promise problem. This new perspective can be applied to arbitrary black-box PRG constructions, and it gives rise to a "non-disjoint" promise problem associated with the black-box PRG construction. For example, the pseudorandom generator construction of [44] induces the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem, which is the problem of distinguishing whether $f \in \mathsf{E}/O(n)$ or $f \notin \mathsf{SIZE}(2^{o(n)})$, given the truth table of a function $f \colon \{0,1\}^n \to \{0,1\}$. Here, $\mathsf{E}/O(n)$ denotes the class of functions that can be computed in time $2^{O(n)}$ with $O(n)$ bits of advice (see Definition 1.10 and Section 4.2 for a precise definition).

There are two types of a pseudorandom generator. One is a *cryptographic* PRG, which is computable in polynomial time in its seed length. This notion is useful for building secure cryptographic primitives. We present in Section 1.2 "non-disjoint" promise problems whose hardness gives rise to a cryptographic hitting set generator. This provides a new approach for establishing the equivalence between the worst-case and average-case complexity of PH. The other is a *complexity-theoretic* PRG, which is allowed to be computed in time exponential in its seed length. This notion is sufficient for the purpose of derandomizing randomized algorithms. In Section 1.3, we generalize the hardness versus randomness framework by using the meta-computational view of black-box PRG constructions, and establish the equivalence between circuit lower bounds for "non-disjoint" promise problems and the existence of hitting set generators. Sections 1.2 and 1.3 can be read independently.

1.2 Worst-case versus average-case complexity of PH

Understanding average-case complexity is a fundamental question in complexity theory. Average-case hardness of NP is a prerequisite for building secure cryptographic primitives, such as one-way functions and cryptographic pseudorandom generators. Indeed, it is not hard to see that if there exists a polynomial-time-computable hitting set generator G, then checking whether a given string is in the image of G is a problem in NP that is hard on average (in the errorless sense). In this section, we present a new approach for proving the average-case hardness of PH, by implicitly constructing a cryptographic hitting set generator.

A fundamental open question in the theory of average-case complexity, pioneered by [51], is to establish the equivalence between the worst-case and average-case complexity of NP.

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

Here, $\mathsf{DistNP} \not\subset \mathsf{AvgP}$ is an average-case analogue of $\mathsf{NP} \neq \mathsf{P}$, and it means that there exist a language $L \in \mathsf{NP}$ and a polynomial-time samplable distribution $\mathcal{D}$ such that no average-case polynomial-time algorithm (equivalently, errorless heuristic scheme) can solve L on instances randomly drawn from $\mathcal{D}$. The reader is referred to the survey of Bogdanov and Trevisan [8] for the formal definitions of DistNP and AvgP as well as background on average-case complexity.

For large enough complexity classes such as PSPACE and EXP, there is a general technique for converting any worst-case hard function f to some two-sided-error average-case hard function $\mathrm{Enc}(f)$ based on error-correcting codes [68, 71]. Here, the encoded function $\mathrm{Enc}(f)$ is computable in PSPACE given oracle access to f; thus, the worst-case and average-case complexities of such large complexity classes are known to be equivalent. Viola [76] showed limitations of such an approach: $\mathrm{Enc}(f)$ cannot be computed in the polynomial-time hierarchy PH^f; thus, the proof technique of using error-correcting codes is not sufficient to resolve Open Question 1.1; it could not even confirm a positive answer to the following weaker open question:

Open Question 1.2. *Does* $\mathsf{PH} \neq \mathsf{P}$ *(or, equivalently,* $\mathsf{P} \neq \mathsf{NP}$*) imply* $\mathsf{DistPH} \not\subset \mathsf{AvgP}$*?*

Observe that Open Question 1.2 is an easier question than Open Question 1.1, since $\mathsf{PH} = \mathsf{P}$ is known to be equivalent to $\mathsf{NP} = \mathsf{P}$.

There are significant obstacles to resolving Open Question 1.1. One is the relativization barrier due to Impagliazzo [43]. Another is the limits of black-box reductions due to Feigenbaum and Fortnow [20] and Bogdanov and Trevisan [8]. The limits of black-box reductions are partially extended to a "barrier" result for Open Question 1.2 in [39].

Recently, a non-black-box worst-case to average-case reduction that is not subject to the latter barrier was presented in [32]. The reduction shows that solving the problem GapMINKT of approximating polynomial-time-bounded Kolmogorov complexity in the worst case can be reduced to solving MINKT on average. We briefly review the definition of GapMINKT below. For an integer $t \in \mathbb{N}$ and an oracle A, a *t-time-bounded A-oracle Kolmogorov complexity*

$K^{t,A}(x)$ of a finite string x is defined as the shortest length of a program that prints x in t steps with oracle access to A (see Section 2 for a precise definition). The promise problem GapMINKT = $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$ asks for approximating $K^t(x)$ within an additive error of $\widetilde{O}(\sqrt{K^t(x)})$, and is formally defined as follows: Π_{YES} consists of $(x, 1^s, 1^t)$ such that $K^t(x) \le s$; and Π_{NO} consists of $(x, 1^s, 1^t)$ such that $K^{\mathsf{poly}(|x|,t)}(x) > s + \widetilde{O}(\sqrt{s})$.

The result of [32] can be seen as providing an approach for establishing the equivalence between worst-case and average-case complexity of NP; indeed, proving NP-hardness of GapMINKT is sufficient for resolving Open Question 1.1 affirmatively. However, the approximation error $\widetilde{O}(\sqrt{s})$ caused by the reduction of [32] is not optimal, which makes the question of proving NP-hardness of GapMINKT potentially harder.

1.2.1 Gap(K^A vs K)

We herein introduce the following "non-disjoint" promise problem.

Definition 1.3. For an oracle A and an approximation quality $\tau\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, the problem $\mathrm{Gap}_\tau(K^A \text{ vs } K)$ is defined as the following promise problem $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$.

$$\Pi_{\text{YES}} := \{\, (x, 1^s, 1^t) \mid K^{t,A}(x) \le s \,\},$$
$$\Pi_{\text{NO}} := \{\, (x, 1^s, 1^t) \mid K^{\tau(|x|,t)}(x) > s + \log \tau(|x|, t) \,\}.$$

By default, we assume that τ is some polynomial and write $\mathrm{Gap}(K^A \text{ vs } K) \in \mathsf{P}$ if there exists some polynomial τ such that $\mathrm{Gap}_\tau(K^A \text{ vs } K) \in \mathsf{P}$.

In this paper, we prove

Theorem 1.4. *Let A be any oracle. If $\mathsf{DistNP}^A \subseteq \mathsf{AvgP}$, then* $\mathrm{Gap}(K^A \text{ vs } K) \in \mathsf{P}$.

An immediate corollary of Theorem 1.4 is an improvement of the reduction of [32], by setting $A := \emptyset$. In particular, in order to resolve Open Question 1.1, it suffices to prove NP-hardness of approximating $K^t(x)$ within an additive error of $\log \tau(|x|, t)$ given $(x, 1^t)$ as input, for any polynomial τ. A key insight for reducing the approximation error is that there are two main sources of the approximation error in the reduction of [32]: One comes from fixing a random coin flip sequence, which we remove by using the (complexity-theoretic) pseudorandom generator constructed by Buhrman, Fortnow, and Pavan [10] under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$. The other comes from the advice complexity of a black-box pseudorandom generator construction, which we reduce by using a "k-wise direct product generator" whose advice complexity is small [35].

More surprisingly, the promise problem is conjectured to be *non-disjoint* for $A := \mathsf{SAT}$. That is, it is conjectured to be *impossible* for any algorithm to solve $\mathrm{Gap}(K^{\mathsf{SAT}} \text{ vs } K)$ — no matter how long the algorithm is allowed to run. Nevertheless, Theorem 1.4 shows that under the assumption that PH is easy on average, there exists a *polynomial-time algorithm* for solving $\mathrm{Gap}(K^{\mathsf{SAT}} \text{ vs } K)$. Taking its contrapositive, this means that, in order to resolve $\mathsf{DistPH} \not\subseteq \mathsf{AvgP}$, it suffices to prove a super-polynomial time lower bound for solving $\mathrm{Gap}(K^{\mathsf{SAT}} \text{ vs } K)$, whose time complexity is conjectured to be *infinity* (in the sense that there exists *no algorithm* that can compute the promise problem).

We now clarify why $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}} \text{ vs } \mathrm{K})$ is conjectured to be non-disjoint. Under the plausible assumption that $\mathsf{E}^{\mathsf{NP}} \neq \mathsf{E}$, it is not hard to see that there are infinitely many strings x such that x is simultaneously a YES and NO instance; here, the string x is defined as the truth table of the characteristic function of $L \in \mathsf{E}^{\mathsf{NP}} \setminus \mathsf{E}/O(n)$ (see Proposition 3.8).

Another corollary of Theorem 1.4 is that under the assumption that $\mathsf{DistPH} \subseteq \mathsf{AvgP}$, any string x that can be compressed with SAT oracle in polynomial time can be also compressed without any oracle. Formally:

Corollary 1.5 (see also Corollary 3.7)**.** *If* $\mathsf{DistPH} \subseteq \mathsf{AvgP}$*, then there exists a polynomial* τ *such that*

$$\mathrm{K}^{\tau(|x|,t)}(x) \leq \mathrm{K}^{t,\mathsf{SAT}}(x) + \log \tau(|x|, t)$$

for any $x \in \{0,1\}^*$ *and* $t \in \mathbb{N}$*.*

Proof Sketch. Under the assumption, $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}} \text{ vs } \mathrm{K})$ can be solved by *some algorithm*. Thus $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}} \text{ vs } \mathrm{K})$ problem must be disjoint, from which the result follows immediately. □

Corollary 1.5 provides a new approach for resolving Open Question 1.2. In order to prove $\mathsf{DistPH} \not\subseteq \mathsf{AvgP}$ under the assumption that $\mathsf{P} \neq \mathsf{NP}$, it suffices to find a string x that can be compressed with SAT oracle but cannot be compressed without SAT oracle. In fact, it suffices to find such a string x under the stronger assumption that $\mathsf{NP} \not\subseteq \mathsf{P}/\mathsf{poly}$. This is because Pavan, Santhanam, and Vinodchandran [60] proved $\mathsf{NP} \not\subseteq \mathsf{P}/\mathsf{poly}$ if the answer to Open Question 1.2 is negative.

More importantly, Theorem 1.4 suggests a more general approach to Open Question 1.2. Note that finding a string x compressible with SAT oracle but incompressible without any oracle corresponds to proving the non-disjointness of $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}} \text{ vs } \mathrm{K})$; this amounts to proving the time complexity of solving $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}} \text{ vs } \mathrm{K})$ is *infinity*. Theorem 1.4 suggests that it suffices to prove that a *polynomial-time* algorithm cannot find a difference between compressible strings under SAT oracle and incompressible strings without any oracle, under the worst-case hardness assumption for NP. In other words, it suffices to prove NP-hardness of $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}} \text{ vs } \mathrm{K})$.

Corollary 1.6 (A new approach for Open Question 1.2)**.** *Suppose that the* $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}} \text{ vs } \mathrm{K})$ *problem is "*NP*-hard under randomized reductions"*[1] *in the sense that*

$$\mathsf{NP} \not\subseteq \mathsf{BPP} \quad \Longrightarrow \quad \mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}} \text{ vs } \mathrm{K}) \notin \mathsf{P}.$$

Then, Open Question 1.2 *is true; that is,*

$$\mathsf{DistPH} \not\subseteq \mathsf{AvgP} \quad \Longleftrightarrow \quad \mathsf{PH} \neq \mathsf{P}.$$

[1]Here we use the weak notion of "NP-hardness" in order to strengthen the result. Corollary 1.6 remains true even if one interprets NP-hardness as a randomized reduction from NP.

In a typical proof of NP-hardness of a disjoint promise problem $\Pi = (\Pi_{\text{Yes}}, \Pi_{\text{No}})$, one needs to carefully design a reduction R from SAT to Π that "preserves" a structure of SAT; i. e., any formula $\varphi \in$ SAT is mapped to $R(\varphi) \in \Pi_{\text{Yes}}$ and any formula $\varphi \in$ UNSAT is mapped to $R(\varphi) \in \Pi_{\text{No}}$. The task of proving NP-hardness of $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$ is potentially much easier: in principle, $R(\varphi)$ can be a fixed input x that is in the intersection of Yes and No instances of $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$; more generally, if a promise problem Π is non-disjoint, then any problem is reducible to Π via a reduction that maps every instance to the intersection of Yes and No instances of Π. It is worth mentioning that proving NP-hardness of $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$ is at least as easy as proving NP-hardness of GapMINKT since GapMINKT is reducible to $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$ via an identity map.

Subsequent to this work, converse directions of Theorem 1.4 and Corollary 1.6 are established in the following sense [33]: DistPH $\subseteq$ AvgP if and only if $\mathrm{Gap}(\mathrm{K}^A$ vs $\mathrm{K}) \in \mathsf{P}$ for every oracle $A \in \mathsf{PH}$. In particular,[2] DistPH $\subseteq$ AvgP $\iff$ PH = P (Open Question 1.2) if and only if $\mathrm{Gap}(\mathrm{K}^A$ vs $\mathrm{K}) \in \mathsf{P}$ for every $A \in \mathsf{PH}$ implies NP $\subseteq$ BPP ("NP-hardness" of $\mathrm{Gap}(\mathrm{K}^A$ vs $\mathrm{K})$). Therefore, the approach to Open Question 1.2 suggested in this work is one of the most general approaches.

1.2.2 Non-NP-hardness results do not apply

A line of work presented evidence that NP-hardness of MINKT is not likely to be established under deterministic reductions (e. g., [49, 54, 40, 38]). For example, it is not hard to see that the proof technique of Murray and Williams [54] (who proved a similar result for MCSP) can be extended to the case of GapMINKT.

Theorem 1.7 (Essentially in [54]; see [35]). *If* GapMINKT *is* NP*-hard under many-one deterministic reductions, then* EXP $\neq$ ZPP.

This result suggests that establishing NP-hardness of GapMINKT under deterministic reductions is a challenging task. In contrast, we observe that a similar "non-NP-hardness" result cannot be applied to the "non-disjoint" promise problem.

Proposition 1.8. *Assume that* NP*-hardness of* $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$ *under many-one reductions implies* EXP $\neq$ ZPP*. Then,* EXP $\neq$ ZPP *holds unconditionally.*

The reason is that $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$ is well defined only if EXP $\neq$ ZPP. More formally:

Proof. There are two cases. Either $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$ is disjoint or not disjoint. In the former case, by Proposition 3.8, we have $\mathsf{E}^{\mathsf{NP}} = \mathsf{E}$, which implies $\mathsf{EXP}^{\mathsf{NP}} = \mathsf{EXP}$ by a standard padding argument; thus, we obtain $\mathsf{EXP} = \mathsf{EXP}^{\mathsf{NP}} = \mathsf{ZPEXP} \neq \mathsf{ZPP}$, where the last inequality follows from a standard translation argument (see, e. g., [11]). In the latter case, there exists a string x that is simultaneously a Yes and No instance. A reduction that always maps to x defines a many-one reduction from any problem to $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$; thus, EXP $\neq$ ZPP follows from the assumption. $\square$

[2] using the fact that P = BPP if DistPH $\subseteq$ AvgP [10]

In light of Proposition 1.8, we leave as an interesting open question whether there is any barrier explaining the difficulty of proving NP-hardness of the non-disjoint promise problem. We mention that $\mathrm{GapMINKT}^{\mathsf{SAT}}$, which is equivalent to $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K}^{\mathsf{SAT}})$, is known to be DistNP-hard [35]. In particular, since $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K}^{\mathsf{SAT}})$ is reducible to $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$ via an identity map, the latter is also DistNP-hard. Therefore, in order to present a barrier for proving NP-hardness of $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$, one must exploit a property that holds for NP but does not hold for DistNP (unless the notion of reducibility is too strong).

1.2.3 $\mathrm{Gap}(F \text{ vs } F^{-1})$: Meta-computational view of HILL's PRG

We also propose another approach towards Open Question 1.1, by introducing a promise problem which asks for distinguishing whether a given function is computable by small circuits, or cannot be inverted by small circuits. Specifically, for an approximation quality τ, we define the promise problem $\mathrm{Gap}_\tau(F \text{ vs } F^{-1})$ as follows. Given a size parameter $s \in \mathbb{N}$ and an integer $n \in \mathbb{N}$ and random access to a function $F\colon \{0,1\}^n \to \{0,1\}^n$, the task is to distinguish the following two cases:

YES: F is computable by a circuit of size s.

NO: F cannot be inverted on average by any F-oracle circuit of size $\tau(n,s)$.

We show that "NP-hardness" of $\mathrm{Gap}_\tau(F \text{ vs } F^{-1})$ for every polynomial τ is enough for resolving Open Question 1.1. More specifically, we prove

Theorem 1.9. *If* DistNP $\subseteq$ AvgP, *then there exist a polynomial τ and a* coRP*-type randomized algorithm that solves $\mathrm{Gap}_\tau(F \text{ vs } F^{-1})$ on input (n,s) in time* poly(n,s). *In particular, Open Question 1.1 is true if $\mathrm{Gap}_\tau(F \text{ vs } F^{-1})$ is "*NP*-hard" for every polynomial τ in the following sense:* NP $\subseteq$ BPP *follows from the assumption that $\mathrm{Gap}_\tau(F \text{ vs } F^{-1})$ admits a* coRP*-type algorithm.*

This is proved by viewing the black-box PRG construction based on any one-way function, which is given by Håstad, Impagliazzo, Levin, and Luby [30], from the meta-computational perspective. Our general proof strategy is given in Section 1.4.

It is easy to observe that $\mathrm{Gap}(F \text{ vs } F^{-1})$ is non-disjoint under the existence of a one-way function, which is one of the most standard cryptographic primitives. Thus, it is widely believed that $\mathrm{Gap}(F \text{ vs } F^{-1})$ is *impossible* to solve. Nevertheless, NP-hardness of $\mathrm{Gap}(F \text{ vs } F^{-1})$ is sufficient for resolving Open Question 1.1.

1.3 Meta-computational circuit lower-bound problems; MCLPs

We now turn our attention to complexity-theoretic hitting set generators. A standard approach for constructing complexity-theoretic pseudorandom generators is to use the hardness versus randomness framework, which reduces the task of constructing a pseudorandom generator to the task of finding an explicit hard function, such as $f \in \mathsf{E} \setminus \mathsf{SIZE}(2^{o(n)})$.

It is, however, a widely accepted fact that proving a circuit size lower bound for an explicit function is extremely hard. Here by an *explicit* function, we mean that a function is computable in $\mathsf{E} = \mathsf{DTIME}(2^{O(n)})$. It is an open question whether there exists an exponential-time-computable function $f \in \mathsf{E}$ that cannot be computed by any circuit of size $4n$ (see [21]). On the other hand, a simple counting argument shows that most functions $f: \{0,1\}^n \to \{0,1\}$ cannot be computed by circuits of size $2^{\alpha n}$ for any constant $\alpha < 1$.

Why is it so difficult to prove a circuit lower bound for an explicit function? We propose to view this question from a *meta-computational* perspective. The fact that it is difficult for *human beings* to show that an explicit function cannot be computed by small circuits suggests that it should be also difficult for *algorithms* to analyze a circuit lower bound. Our results indicate that if we can make this intuition formal, then we get breakthrough results in complexity theory.

Specifically, we herein introduce a family of new computational problems, which we call *Meta-computational Circuit Lower-bound Problems* (MCLPs). These problems ask for distinguishing the truth table of explicit functions from hard functions. For example, we propose the following promise problem:

The E vs $\mathsf{SIZE}(2^{o(n)})$ **Problem** (informal)[3]

> Given the truth table of a function $f: \{0,1\}^n \to \{0,1\}$, distinguish whether $f \in \mathsf{E}/O(n)$ or $f \notin \mathsf{SIZE}(2^{o(n)})$.

Before defining the problem formally, let us first observe that the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem is closely related to the open question of whether $\mathsf{E} \not\subset \mathsf{SIZE}(2^{o(n)})$. Indeed, it is not hard to show that $\mathsf{E}/O(n) \not\subset \mathsf{SIZE}(2^{o(n)})$ if and only if $\mathsf{E} \not\subset \mathsf{SIZE}(2^{o(n)})$ by regarding an advice string as a part of input. Therefore, the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem is non-disjoint if and only if $\mathsf{E} \not\subset \mathsf{SIZE}(2^{o(n)})$.

We now define the problem formally. According to the standard notion of advice [46], the complexity class $\mathsf{E}/O(n)$ is defined as a subset of functions $f: \{0,1\}^* \to \{0,1\}$ that are defined on all the strings of any length. Thus, "$f \in \mathsf{E}/O(n)$" does not make sense for a function $f: \{0,1\}^n \to \{0,1\}$. Instead, we interpret advice by using the notion of Levin's resource-bounded Kolmogorov complexity [50] so that the notion of advice is meaningful for a finite function $f: \{0,1\}^n \to \{0,1\}$. For a string $x \in \{0,1\}^*$, let $\mathrm{Kt}(x)$ denote the Levin's Kolmogorov complexity of a string x, which is defined as the minimum of $|M| + \log t$ over all the programs M that output x in time t; here, $|M|$ denotes the description length of M. The E vs $\mathsf{SIZE}(2^{o(n)})$ Problem is formally defined as follows.

Definition 1.10. For any functions $t, s: \mathbb{N} \to \mathbb{N}$, let $(\Pi_{\text{YES}}(t(n)), \Pi_{\text{NO}}(s(n)))$ denote the promise problem defined as

$$\Pi_{\text{YES}}(t(n)) := \{\, f \in \{0,1\}^{2^n} \mid \mathrm{Kt}(f) \le \log t(n), n \in \mathbb{N} \,\},$$
$$\Pi_{\text{NO}}(s(n)) := \{\, f \in \{0,1\}^{2^n} \mid \mathrm{size}(f) > s(n), n \in \mathbb{N} \,\}.$$

Here, we identity a function $f: \{0,1\}^n \to \{0,1\}$ with its truth table representation $f \in \{0,1\}^{2^n}$, and $\mathrm{size}(f)$ denotes the minimum size of a circuit that computes f.

[3] This problem may be called the $\mathsf{E}/O(n)$ vs $\mathsf{SIZE}(2^{o(n)})$ Problem; however, for the sake of notational simplicity (and for the reason described in the following paragraph), we omit "$/O(n)$" from the name of the problem.

The E vs $\mathsf{SIZE}(2^{o(n)})$ *Problem* is defined as the family $\{(\Pi_{\text{Yes}}(2^{cn}), \Pi_{\text{No}}(2^{\alpha n}))\}_{c,\alpha>0}$ of the promise problems. A family $\{\Pi\}$ of problems is said to be *solved* by a class $\mathfrak{C}$ and denoted by $\{\Pi\} \in \mathfrak{C}$ if every problem in the family is solved by some algorithm in $\mathfrak{C}$.

The idea behind Definition 1.10 is that the complexity class $\mathsf{E}/O(n)$ can be characterized as the class of the functions $f = \{f_n : \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$ such that, for some constant c, for all large $n \in \mathbb{N}$, $\mathrm{Kt}(f_n) \le cn$ holds. Indeed, $f \in \mathsf{E}/O(n)$ means that the truth table of f_n can be described by a Turing machine of description length $O(n)$ in time $2^{O(n)}$ for all large n. The relationship between complexity classes with advice and resource-bounded Kolmogorov complexity will be explained in detail in Section 4.2, where we interpret "$\mathsf{DTIME}(t(n))/a(n)$" as a subset of functions $f\colon \{0,1\}^n \to \{0,1\}$.

1.3.1 Meta-computational view of the hardness vs randomness framework

We show that a nearly linear-size $\mathsf{AC}^0 \circ \mathsf{XOR}$ circuit size lower bound for solving the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem exactly characterizes the existence of a hitting set generator secure against $\mathsf{AC}^0 \circ \mathsf{XOR}$. Here, $\mathsf{AC}^0 \circ \mathsf{XOR}$ denotes the class of constant-depth circuits that consist of unbounded-fan-in AND, OR and XOR gates, and XOR gates are allowed to be present only in the bottom later.

Theorem 1.11. *The following (Items 1 to 4) are equivalent.*

1. *There exists a hitting set generator $G = \{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ computable in time $n^{O(1)}$ and secure against linear-size $\mathsf{AC}^0 \circ \mathsf{XOR}$ circuits.*

2. *For all large $N \in \mathbb{N}$, there exists no $\mathsf{AC}^0 \circ \mathsf{XOR}$ circuit of size $N^{1+o(1)}$ that computes the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem, where $N = 2^n$ denotes the input length.*

The condition can be equivalently stated without referring to the "non-disjoint" promise problem. Let $\mathrm{MKtP}[O(\log N), N^{o(1)}]$ denote the family of the promise problems $\mathrm{MKtP}[c\log N, N^{\alpha}]$ for constants $c, \alpha > 0$; here, for functions $s, t\colon \mathbb{N} \to \mathbb{N}$, $\mathrm{MKtP}[s(N), t(N)]$ denotes the promise problem of deciding whether $\mathrm{Kt}(x) \le s(|x|)$ or $\mathrm{Kt}(x) > t(|x|)$ on input x. Then, the following are equivalent as well.

3. *For all large $N \in \mathbb{N}$, there exists no $\mathsf{AC}^0 \circ \mathsf{XOR}$ circuit of size $N^{1+o(1)}$ that computes* $\mathrm{MKtP}[O(\log N), N^{o(1)}]$.

4. *For any constant $k \in \mathbb{N}$, for all large $N \in \mathbb{N}$, there exists no $\mathsf{AC}^0 \circ \mathsf{XOR}$ circuit of size N^k that computes* $\mathrm{MKtP}[O(\log N), N^{o(1)}]$.

Observe that Item 1 of Theorem 1.11 implies a strongly exponential AC^0 circuit lower bound for E (i. e., E cannot be computed by AC^0 circuits of size $2^{o(n)}$), which also implies that $\mathsf{EXP} \not\subset \mathsf{NC}^1$ (see, e. g., [3, 58, 28]). The current best AC^0 circuit lower bound for E is due to Håstad [29], who proved that the Parity function on n inputs cannot be computed by depth-d AC^0 circuits of size $2^{o(n^{1/(d-1)})}$. Theorem 1.11 shows that, in order to improve this state-of-the-art lower bound, it is sufficient to prove a *nearly linear* $\mathsf{AC}^0 \circ \mathsf{XOR}$ lower bound for the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem. In contrast, the minimum circuit for computing the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem is *infinity* under the standard circuit lower bound assumption that $\mathsf{E} \not\subset \mathsf{SIZE}(2^{o(n)})$.

It is instructive to compare our results with the hardness versus randomness framework. In order to obtain a hitting set generator in the latter framework, we need to find an explicit function that is hard for small circuits to compute. In our framework, finding an explicit hard function corresponds to proving that the minimum circuit size for computing MCLPs is *infinity* (or, in other words, proving that there exists no circuit of *any size* that computes MCLPs[4]). Our results significantly weaken the assumption needed to obtain a hitting set generator: It suffices to show that a *nearly linear* circuit cannot find the difference between an explicit function and a hard function.

Our results can be also stated based on the case analysis. There are two cases. (1) When the circuit lower bound that $\mathsf{E} \not\subset \mathsf{SIZE}(2^{o(n)})$ holds, the work of [44] already implies the existence of a pseudorandom generator. (2) Even if the circuit lower bound does fail, Theorem 1.11 shows that a very modest $\mathsf{AC}^0 \circ \mathsf{XOR}$ lower bound for the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem (which is a disjoint promise problem under the assumption) implies the existence of a hitting set generator. In either case, we obtain a hitting set generator. Our results generalize the hardness versus randomness framework in this sense.

Previously, based on the hardness versus randomness framework, it is known that $\mathsf{E} \not\subset \mathfrak{C}$ is equivalent to the existence of a pseudorandom generator secure against $\mathfrak{C}$ for a sufficiently large class $\mathfrak{C}$ (see, e. g., [23]). However, in the previous approach, one needs to transform a worst-case $\mathfrak{C}$-circuit lower bound to an average-case $\mathfrak{C}$-circuit lower bound; thus $\mathfrak{C}$ needs to be a sufficiently large so that it can perform local decoding, which requires the majority gate [66]. For any circuit class $\mathfrak{C}$ smaller than TC^0, it was not clear whether the existence of a hitting set generator secure against $\mathfrak{C}$ is equivalent to some worst-case $\mathfrak{C}$-circuit lower bound. Theorem 1.11 establishes the first equivalence for the circuit class $\mathfrak{C} = \mathsf{AC}^0 \circ \mathsf{XOR}$, which is smaller than TC^0 [62].

Our results can be stated without the non-standard notion of promise problem, as in Items 3 and 4 of Theorem 1.11. Indeed, any promise problem in the family $\mathrm{MKtP}[O(\log N), N^{o(1)}]$ asks for approximating the Kt-complexity of a given string, and it is always a disjoint promise problem. In our terminology, $\mathrm{MKtP}[O(\log N), N^{o(1)}]$ is equivalent to the E vs $\mathsf{DTIME}(2^{2^{o(n)}})/2^{o(n)}$ Problem. Since $\mathsf{SIZE}(2^{o(n)}) \subseteq \mathsf{DTIME}(2^{2^{o(n)}})/2^{o(n)}$, one can observe that the E vs $\mathsf{DTIME}(2^{2^{o(n)}})/2^{o(n)}$ Problem is reducible to the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem via an identity map, which explains the implication from Item 3 to Item 2 in Theorem 1.11.

We mention that it is not hard to prove an AC^0 lower bound for $\mathrm{MKtP}[O(\log N), N^{o(1)}]$ and in particular for the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem.

Proposition B.1. *For any constants $\alpha < 1, k, d \in \mathbb{N}$, there exists a constant c such that*

$$\mathrm{MKtP}[c \log N, N^{\alpha}] \notin \mathsf{i.o.AC}^0_d(N^k).$$

This can be proved by using the pseudorandom restriction method as in [37, 18, 14]. (See Appendix B for a proof.) Extending Proposition B.1 to the case of $\mathsf{AC}^0 \circ \mathsf{XOR}$ is sufficient for a

[4]This should be compared with the fact that any *disjoint* promise problem can be computed by a circuit of size $O(2^n/n)$ on inputs of length n.

breakthrough result in light of Theorem 1.11.

For any classes $\mathfrak{C}, \mathfrak{D}$ of functions, one can define the $\mathfrak{C}$ vs $\mathfrak{D}$ Problem. A particularly interesting problem is the E vs $\widetilde{\mathsf{AC}^0}(2^{o(n)}; \frac{1}{2} - 2^{-o(n)})$ Problem, where $\widetilde{\mathfrak{D}}(s;\delta)$ denotes the class of functions that can be computed by a $\mathfrak{D}$-circuit of size s on at least a $(1-\delta)$ fraction of inputs. We prove that, if nearly linear-size AC^0 circuits cannot distinguish an explicit function from a function that cannot be approximated by small AC^0 circuits, then a logarithmic-seed-length hitting set generator can be obtained. (Moreover, the converse direction is easy to prove.)

Theorem 1.12. *The following are equivalent.*

1. *For all large $N = 2^n$, the E vs $\widetilde{\mathsf{AC}^0}(2^{o(n)}; \frac{1}{2} - 2^{-o(n)})$ Problem cannot be decided by AC^0 circuits of size $N^{1+o(1)}$.*

2. *There exists a hitting set generator $G = \{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ computable in time $n^{O(1)}$ and secure against linear-size AC^0 circuits.*

3. $\mathrm{MKtP}[O(\log N), N-1] \notin \mathsf{i.o.AC}^0(N^{1+\beta})$ *for some constant $\beta > 0$.*

4. $\mathrm{MKtP}[O(\log N), N-1] \notin \mathsf{i.o.AC}^0(N^k)$ *for any constant k.*

An interesting aspect of Theorem 1.12 is its self-referential nature; intuitively, Item 1 means that AC^0 circuits cannot analyze AC^0 circuits itself. Note that self-reference is crucial for proving, e. g., time hierarchy theorems for uniform computational models. Theorem 1.12 provides an analogue in a non-uniform circuit model.

Why do we consider "non-disjoint" promise problems, despite the fact that Theorems 1.11 and 1.12 can be stated without referring to the non-standard notion? A couple of remarks are in order. First, Theorem 1.11 is obtained by viewing (a variant of) the black-box PRG construction of Impagliazzo and Wigderson [44] from a meta-computational perspective; thus, it is natural to state Theorem 1.11 as a connection between the existence of a hitting set generator and a lower bound for the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem. Second, an identity map reduces $\mathrm{MKtP}[O(\log N), N^{o(1)}]$ to the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem, and thus it is easier to prove a lower bound for the latter problem. Third, the known worst-case-and-average-case equivalence between $\mathsf{E} \subseteq \mathsf{SIZE}(2^{o(n)})$ and $\mathsf{E} \subseteq \widetilde{\mathsf{SIZE}}(2^{o(n)}; \frac{1}{2} - 2^{-o(n)})$ [68] can be naturally regarded as a reduction from the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem to the E vs $\widetilde{\mathsf{SIZE}}(2^{o(n)}; \frac{1}{2} - 2^{-o(n)})$ Problem. Indeed, Theorem 1.12 is proved by viewing the Nisan–Wigderson pseudorandom generator from a meta-computational perspective, and then Theorem 1.12 is translated into Theorem 1.11 by using the worst-case-and-average-case equivalence. Lastly and most importantly, in the other settings, the non-disjointness itself provides new consequences, such as Corollaries 1.5 and 1.16.

We also present a potential approach for resolving the $\mathsf{RL} = \mathsf{L}$ question. Here, RL is the complexity class of languages that can be solved by a one-sided-error randomized $O(\log n)$-space Turing machine that reads its random tape *only once*. A canonical approach for proving $\mathsf{RL} = \mathsf{L}$ is to construct a log-space-computable hitting set generator of seed length $O(\log n)$ secure against $O(n)$-size read-once branching programs. State-of-the-art results are the pseudorandom generator of seed length $O(\log^2 n)$ by Nisan [55] for read-once (known-order) oblivious branching programs, and the pseudorandom generator of seed length $O(\log^3 n)$ by Forbes and Kelley [22] for read-once unknown-order oblivious branching programs.[5]

We show that a hitting set generator of seed length $O(\log n)$ can be constructed if nearly linear-size read-once co-nondeterministic branching programs cannot distinguish linear-space-computable functions from hard functions.

Theorem 1.13. *Suppose that, for some constants $\beta, \delta \in (0, 1/2)$, the* $\mathsf{DSPACE}(n)$ vs $\widetilde{\mathsf{SIZE}}(2^{o(n)}; \delta)$ *Problem cannot be computed by read-once co-nondeterministic branching programs of size $N^{1+\beta}$ for all large input lengths $N = 2^n$. Then there exists a hitting set generator $G = \{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ computable in $O(\log n)$ space and secure against linear-size read-once branching programs (and, in particular,* $\mathsf{RL} = \mathsf{L}$ *follows; moreover,* $\mathsf{BPL} = \mathsf{L}$ *holds*[6]*).*

Theorem 1.13 can be compared with a result of Klivans and van Melkebeek [48]. Based on the hardness versus randomness framework, they showed $\mathsf{BPL} = \mathsf{L}$ under the assumption that $\mathsf{DSPACE}(n)$ requires circuits of size $2^{\Omega(n)}$. Under the same assumption, by using a worst-case-to-average-case reduction for $\mathsf{DSPACE}(n)$, it can be shown that the $\mathsf{DSPACE}(n)$ vs $\widetilde{\mathsf{SIZE}}(2^{o(n)}; \delta)$ Problem is non-disjoint for any constant $\delta < 1/2$ (see Proposition 4.11). In this case, the minimum size of a co-nondeterministic branching program for computing the MCLP is *infinity*; thus, Theorem 1.13 generalizes the result of [48].

It should be noted that the limits of the computational power of read-once non-deterministic branching programs are "well understood." For example, Borodin, Razborov, and Smolensky [9] presented an explicit function that cannot be computed by any read-k-times nondeterministic branching program of size $2^{o(n)}$ for any constant k. Theorem 1.13 shows that, in order to resolve $\mathsf{BPL} = \mathsf{L}$, it suffices to similarly analyze the read-once co-nondeterministic branching program size for computing the MCLP. This approach could be useful; by using the Nechiporuck method, it can be shown that neither nondeterministic nor co-nondeterministic branching programs of size $o(N^{1.5}/\log N)$ can compute MKtP [17], which is a much more general lower bound than read-k-times nondeterministic branching programs.

We also mention that a partial converse of Theorem 1.13 is easy to prove: If there exists a log-space-computable hitting set generator secure against linear-size read-once nondeterministic branching programs, then the $\mathsf{DSPACE}(n)$ vs $\widetilde{\mathsf{SIZE}}(2^{o(n)}; \delta)$ Problem cannot be computed by a read-once co-nondeterministic branching programs of size N^k, where $N = 2^n$ and $\delta < 1/2$ is an arbitrary constant. More generally, any results showing the existence of a hitting set generator secure against $\mathfrak{C}$ must entail a $\mathsf{co}\mathfrak{C}$-lower bound for MCLPs (see Proposition 4.22).

1.3.2 Non-trivial derandomization and lower bounds for MKtP

Our proof techniques can be also applied to *uniform* algorithms. We consider the question of whether one-sided-error uniform algorithms can be non-trivially derandomized in time

[5]In the area of unconditional derandomization of space-bounded randomized algorithms, it is common to assume that a branching program is oblivious and reads the input in the fixed order. Here, we do not assume these properties.

[6]This is because of a recent result of Cheng and Hoza [16], which shows $\mathsf{BPL} = \mathsf{L}$ from the existence of a hitting set generator.

$2^{n-\omega(\sqrt{n}\log n)}$. We say that an algorithm A is a *derandomization algorithm for* $\mathsf{DTIME}(t(n))$ if, for any machine M running in time $t(n)$, A takes 1^n and a description of M as input and outputs $y \in \{0,1\}^n$ such that $M(y) = 1$ for infinitely many $n \in \mathbb{N}$ such that $\Pr_{x\sim\{0,1\}^n}[M(x) = 1] \geq \frac{1}{2}$. Unlike the standard setting of a derandomization algorithm for non-uniform computational models, the description length of M is at most a constant independent of input length n; thus, our notion of derandomization algorithm is essentially equivalent to the existence of a hitting set generator secure against $\mathsf{DTIME}(t(n))$. Applying our proof techniques to this setting, we establish the following equivalence between the existence of a derandomization algorithm for uniform algorithms and a lower bound for approximating Kt complexity.

Theorem 1.14 (informal; see Theorem 5.5)**.** *For any constant* $0 < \epsilon < 1$*, the following are equivalent:*

1. *There exists a derandomization algorithm for* $\mathsf{DTIME}(2^{O(\sqrt{N}\log N)})$ *that runs in time* $2^{N-\omega(\sqrt{N}\log N)}$.
2. $\mathrm{MKtP}[N - \omega(\sqrt{N}\log N), N-1] \notin \mathsf{DTIME}(2^{O(\sqrt{N}\log N)})$.
3. $\mathrm{MKtP}[N^\epsilon, N^\epsilon + \omega(\sqrt{N^\epsilon}\log N)] \notin \mathsf{DTIME}(2^{O(\sqrt{N^\epsilon}\log N)})$.

Usually, the time complexity is measured with respect to the input size. Our result, however, suggests that the time complexity of $\mathrm{MKtP}[s(N), s(N) + \omega(\sqrt{N}\log N)]$ is well captured by the size parameter $s(N)$ rather than the input size N: Indeed, Theorem 1.14 implies that

$$\mathrm{MKtP}[N^\epsilon, N^\epsilon + \omega(\sqrt{N^\epsilon}\log N)] \in \mathsf{DTIME}(2^{O(\sqrt{N^\epsilon}\log N)})$$

is equivalent to

$$\mathrm{MKtP}[N^\delta, N^\delta + \omega(\sqrt{N^\delta}\log N)] \in \mathsf{DTIME}(2^{O(\sqrt{N^\delta}\log N)})$$

for any $0 < \epsilon, \delta < 1$.

Theorem 1.14 highlights the importance of a lower bound for MKtP. In fact, it is a long-standing open question whether $\mathrm{MKtP} \notin \mathsf{P}$, despite the fact that MKtP is an EXP-complete problem under non-uniform reductions [2]. Towards resolving this open question, we present several results that might be useful. We show that some "non-disjoint" promise problem can be solved under the assumption that $\mathrm{MKtP} \in \mathsf{P}$.

Theorem 1.15. *Assume that* $\mathrm{MKtP} \in \mathsf{P}$*. Then, there exists a* $2^{\widetilde{O}(\sqrt{n})}$*-time algorithm that solves the* Kt vs K^t *Problem, which is defined as follows: Given a string* $x \in \{0,1\}^*$ *of length n and a parameter* $s \in \mathbb{N}$*, distinguish whether* $\mathrm{Kt}(x) \leq s$ *or* $\mathrm{K}^t(x) \geq s + O(\sqrt{s}\log n + \log^2 n)$*, where* $t := n^{\log n}$.

Using the disjointness of the Kt vs K^t Problem and setting $s := \mathrm{Kt}(x)$, we obtain

Corollary 1.16. *If* $\mathrm{MKtP} \in \mathsf{P}$*, then* $\mathrm{K}^t(x) \leq \mathrm{Kt}(x) + O(\sqrt{\mathrm{Kt}(x)}\log n + \log^2 n)$ *for any* $x \in \{0,1\}^n$*, where* $t := n^{\log n}$.

Since $\mathrm{Kt}(x) \leq \mathrm{K}^t(x) + O(\log^2 n)$ holds for $t := n^{\log n}$ unconditionally, Corollary 1.16 shows that $\mathrm{Kt}(x)$ and $\mathrm{K}^t(x)$ are equal up to an additive term of $\widetilde{O}(\sqrt{n})$ under the assumption that

MKtP $\in$ P. In particular, in order to resolve MKtP $\notin$ P, it suffices to prove that $\mathrm{K}^t(x)$ cannot be approximated within an additive error $\widetilde{O}(\sqrt{n})$ in polynomial time. It was shown in [35] that there is no polynomial-time algorithm that computes $\mathrm{K}^t(x)$ *exactly* on input x. The large additive error term $\widetilde{O}(\sqrt{n})$ prevents us from combining the lower bound proof technique of [35] and Corollary 1.16.

1.3.3 Related work: Minimum Circuit Size Problem

The definitions of MCLPs are inspired by the *Minimum Circuit Size Problem* (MCSP). While the history of MCSP is said to date back to as early as 1950s [69], its importance was not widely recognized until Kabanets and Cai [45] named the problem as MCSP and investigated it based on the natural proof framework of Razborov and Rudich [63]. The task of MCSP is to decide whether there exists a size-s circuit that computes f, given the truth table of a function $f\colon \{0,1\}^n \to \{0,1\}$ and a size parameter $s \in \mathbb{N}$. It turned out that MCSP is one of the central computational problems in relation to wide research areas of complexity theory, including circuit lower bounds [63, 58], learning theory [13], cryptography [63, 2, 4, 65], and average-case complexity [32].

It has been recognized that MCSP lacks one desirable mathematical property — *monotonicity* with respect to an underlying computational model. MCSP can be defined for any circuit classes $\mathfrak{C}$; for example, $\mathfrak{C}$-MCSP stands for a version of MCSP where the task is to find the minimum $\mathfrak{C}$-circuit size; MCSP^A stands for the minimum A-oracle circuit size problem. We are tempted to conjecture that, as a computational model becomes stronger, the corresponding minimization problem becomes harder; e. g., MCSP^A should be harder than MCSP for any oracle A. However, this is not the case — Hirahara and Watanabe [38] showed that there exists an oracle A such that MCSP $\not\leq_T^p$ MCSP^A unless MCSP $\in$ P. Moreover, DNF-MCSP [52, 3] and (DNF $\circ$ XOR)-MCSP [36] are known to be NP-complete, whereas NP-completeness of MCSP is a long-standing open question.

Why does the monotonicity of MCSP fail? For two circuit classes $\mathfrak{C}$ and $\mathfrak{D}$, define the $\mathfrak{C}$ vs $\mathfrak{D}$ Problem to be the promise problem of distinguishing whether f has a $\mathfrak{C}$-circuit of size s or f does not have a $\mathfrak{D}$-circuit of size s, given the truth table of a function $f\colon \{0,1\}^n \to \{0,1\}$ and a size parameter s. Then, $\mathfrak{C}$-MCSP can be regarded as a special case of the $\mathfrak{C}$ vs $\mathfrak{D}$ Problem where $\mathfrak{C} = \mathfrak{D}$. It is easy to observe that the $\mathfrak{C}$ vs $\mathfrak{D}$ Problem is reducible to the $\mathfrak{C}'$ vs $\mathfrak{D}'$ Problem via an identity map if $\mathfrak{C} \subseteq \mathfrak{C}'$ and $\mathfrak{D} \supseteq \mathfrak{D}'$; thus, MCLPs have monotonicity properties in this sense. In contrast, the monotonicity property of MCLPs fails when $\mathfrak{C} = \mathfrak{D} \subsetneq \mathfrak{C}' = \mathfrak{D}'$, which corresponds to the case of MCSP.

In an attempt to remedy the monotonicity issue, Hirahara and Santhanam [37] observed that average-case complexity of MCSP is monotone increasing. Carmosino, Impagliazzo, Kabanets, and Kolokolova [13] implicitly showed that the complexity of MCSP is monotone increasing under non-black-box reductions.

In contrast, MCLPs incorporate the monotonicity property in the definition itself, which makes a mathematical theory cleaner. For example, recall that it can be shown that $\mathsf{E} \not\subset \mathsf{SIZE}(2^{o(n)})$ if and only if $\mathsf{E} \not\subset \widetilde{\mathsf{SIZE}}(2^{o(n)}; \frac{1}{2} - 2^{-o(n)})$ by using error-correcting codes [68]. Viewing this equivalence from a meta-computational perspective, it can be interpreted as an efficient reduction from the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem to the E vs $\widetilde{\mathsf{SIZE}}(2^{o(n)}; \frac{1}{2} - 2^{-o(n)})$ Problem (see Theorem 4.27). A similar reduction was presented for MCSP under the assumption that $\mathsf{EXP} \subseteq \mathsf{P/poly}$ [14]; finding such a reduction for MCSP requires certain creativity, whereas working with the definition of MCLPs makes it trivial to find the reduction.

1.3.4 Related work: Hardness Magnification

A recent line of work [58, 57, 53, 15, 14] exhibit surprising phenomena, which are termed as "hardness magnification phenomena." Oliveira, Santhanam, and Pich [58, 57] showed that very weak lower bounds for MCSP and related problems are sufficient for resolving long-standing open questions about circuit lower bounds, such as $\mathsf{EXP} \not\subseteq \mathsf{NC}^1$. An interesting aspect of hardness magnification phenomena is that, as argued in [5, 58], the argument does not seem to be subject to the natural proof barrier of Razborov and Rudich [63], which is one of the major obstacles of complexity theory. Our results can be seen as a new interpretation of hardness magnification phenomena from the perspective of a construction of a hitting set generator.

It is worthwhile to point out that our reductions have very similar structures to hardness magnification phenomena. For example, it was shown in [58, 57] that, for a parameter $s = s(N)$, the problem $\mathrm{MKtP}[s, s + O(\log N)]$ can be solved by an $\mathsf{AND}_{O(N)} \circ D_{\mathsf{poly}(s)} \circ \mathsf{XOR}$ circuit, where $D_{\mathsf{poly}(s)}$ is an oracle gate computable in EXP that takes an input of length $\mathsf{poly}(s)$. This reduction shows that $\mathsf{EXP} \subseteq \mathsf{P/poly}$ implies $\mathrm{MKtP}[s, s + O(\log N)] \in \mathsf{SIZE}(N \cdot \mathsf{poly}(s))$. Taking its contrapositive, it can be interpreted as a hardness magnification phenomenon: for $s(N) \ll N$, a (seemingly) weak lower bound $\mathrm{MKtP}[s, s + O(\log N)] \notin \mathsf{SIZE}(N \cdot \mathsf{poly}(s))$ can be "hardness magnified" to a strong circuit lower bound $\mathsf{EXP} \not\subseteq \mathsf{P/poly}$.

Theorem 1.11 has exactly the same structure with the reduction mentioned above. Given a circuit $D_{N^{o(1)}}$ that avoids a hitting set generator, we construct a nearly linear-size $\mathsf{AND}_{O(N)} \circ D_{N^{o(1)}} \circ \mathsf{XOR}$ circuit that computes the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem. Thus, our reductions are as efficient as the reductions presented in the line of work of hardness magnification phenomena.

More importantly, our results significantly strengthen the consequences of hardness magnification: Not only circuit lower bounds, but also hitting set generators can be obtained. This is especially significant for the case of read-once branching programs. Since there is already an exponential size lower bound for read-once branching programs [9], it does not make sense to try to hardness-magnify a lower bound for read-once branching programs. In contrast, our results (Theorem 1.13) indicate that a nearly linear size lower bound for co-nondeterministic read-once branching programs is enough for resolving $\mathsf{BPL} = \mathsf{L}$.

An intriguing question about hardness magnification is this: By using hardness magnification phenomena, can we prove any new consequences, such as circuit lower bounds or derandomization? Theorem 1.13 adds a new computational model, i. e., co-nondeterministic read-once branching programs, for exploring this question.

Chen, Hirahara, Oliveira, Pich, Rajgopal, and Santhanam [14] proposed a barrier for the question, termed as a "locality barrier." Briefly speaking, the idea there is to regard hardness magnification phenomena as a (black-box) reduction to oracles with small fan-in, and then to show that most circuit lower bound proofs can be extended to rule out such a reduction; thus, such a circuit lower bound proof technique cannot be combined with hardness magnification phenomena. A salient feature of our reductions is that our reductions are *non-black-box* in the sense that we exploit the efficiency of oracles; the non-black-box property appears in the definition of No instances of the $\mathfrak{C}$ vs $\mathfrak{D}$ Problem. In light of this, our results provide a potential approach for bypassing the locality barrier: Try to develop a circuit lower bound proof technique that crucially exploits the structure of the No instances of the $\mathfrak{C}$ vs $\mathfrak{D}$ Problem. The existing circuit lower bound proof techniques for MCSP and related problems fail to exploit such a structure.

1.4 Proof techniques: Meta-computational view of PRG constructions

All of our results are given by a *single principle* — that views any black-box pseudorandom generator construction from a meta-computational perspective. The differences among our theorems simply originate from the fact that we use a different black-box pseudorandom generator construction. The underlying principle is this:

> Any black-box construction of a pseudorandom generator G^f based on a hard function $f \notin \mathcal{R}$ gives rise to a non-black-box security reduction for a hitting set generator based on the hardness of a non-disjoint promise problem (e. g., the E vs $\mathcal{R}$ Problem).

For the purpose of exposition, we take a specific PRG construction of Impagliazzo and Wigderson [44], and prove a connection between the PRG construction and the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem. Specifically, in this section, we present a self-contained proof of the following result.

Theorem 1.17. *There exists a universal constant β such that the following are equivalent.*

1. *The E vs $\mathsf{SIZE}(2^{o(n)})$ Problem cannot be solved by a circuit of size N^β on inputs of length $N = 2^n$ for all large N.*
2. *There exists a hitting set generator $H = \{H_m : \{0,1\}^{O(\log m)} \to \{0,1\}^m\}_{m\in\mathbb{N}}$ computable in time $m^{O(1)}$ and secure against linear-size circuits.*

The implication from Item 2 to Item 1 is by now a standard fact in the literature of MCSP and Kolmogorov complexity [63, 2]. The implication from Item 1 to Item 2 is shown by applying our principle to the following black-box pseudorandom generator construction of [44].

Lemma 1.18 (Impagliazzo and Wigderson [44])**.** *There is a constant β such that, for any constant $\alpha > 0$, there exist a constant $0 < \alpha' < 1/2$ and a "black-box pseudorandom generator construction" $G^f : \{0,1\}^{O(n)} \to \{0,1\}^{2^{\alpha' n}}$ that takes a function $f : \{0,1\}^n \to \{0,1\}$ for any $n \in \mathbb{N}$ and satisfies the following.*

Explicitness *$G^f(z)$ is computable in time $2^{O(n)}$ given the truth table of f and a seed z.*

Constructivity *For every fixed seed $z \in \{0,1\}^{O(n)}$, there is a circuit C_z of size $2^{(\beta-2)n}$ that takes as input the 2^n-bit truth table of a function f and computes $G^f(z)$.*

Security *For every function* $D\colon \{0,1\}^{2^{\alpha' n}} \to \{0,1\}$ *that* $\frac{1}{4}$-distinguishes[7] *the output distribution of* $G^f(\text{-})$ *from the uniform distribution, then* $f \in \mathsf{SIZE}^D(2^{(\alpha-\alpha')n})$.

Here, by "black-box",[8] we mean that the security of the PRG is proved by a (black-box) reduction, i. e., the security reduction works for *every* function D; otherwise, the reduction is called non-black-box (e. g., in the case when the reduction works only if the oracle is efficient). This is in the same spirit with the non-black-box reduction of [32], which overcomes the black-box reduction barrier of Bogdanov and Trevisan [8]. We explain below how a black-box PRG construction gives rise to a *non-black-box* security reduction of some hitting set generator.

Proof of Theorem 1.17. (Item 1 ⇒ 2) The goal is to construct some secure hitting set generator $H = \{H_m\colon \{0,1\}^{c_0 \log m} \to \{0,1\}^m\}_{m\in\mathbb{N}}$ for some constant c_0. As a choice of H, we simply take a "universal" hitting set generator: Let U be a universal Turing machine, i. e., a machine that simulates every Turing machine efficiently. Then we define $H_m(z)$ to be the output of U on input z if U outputs a string of length m in $2^{|z|}$ steps, where $z \in \{0,1\}^{c_0 \log m}$. The important property of H_m is that every string $x \in \{0,1\}^m$ with $\mathrm{Kt}(x) \le c_0 \log m$ is contained in the image of H_m. (The choice of H is universal, in the sense that the existence of some exponential-time computable HSG implies that H is also secure.)

The strategy for proving the security of a hitting set generator H is to regard a function $f\colon \{0,1\}^n \to \{0,1\}$ as an input of the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem, and to view the black-box PRG construction G^f as a (non-black-box) reduction that proves the security of H.

We claim that H is a hitting set generator secure against linear-size circuits, under the assumption that the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem cannot be solved by small circuits. To this end, we present a non-black-box reduction $R^{(\text{-})}$ from the task of solving the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem to the task of "avoiding" H. That is, we prove the contrapositive of Item 1 ⇒ 2: Assume that H is not secure (for every constant c_0, which decides the seed length of H). Then, for infinitely many $m \in \mathbb{N}$, there exists a linear-size circuit D that *avoids* H_m, i. e., every string in the image of H_m is rejected by D whereas D accepts at least a half of all the m-bit inputs. We claim that there exists a small oracle circuit R^D that solves the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem. The randomized circuit R^D is extremely simple:

A randomized algorithm R^D for solving the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem with D oracle

Given f as an input, pick a random seed z of G^f, and accept if and only if $D(G^f(z)) = 0$.

More formally, recall that the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem is defined as the family of promise problems $\Pi_{c,\alpha} = (\Pi_{\text{YES}}(2^{cn}), \Pi_{\text{NO}}(2^{\alpha n}))$ for every $c, \alpha > 0$ as in Definition 1.10. We need to prove that, for any constants $c, \alpha > 0$, for infinitely many $n \in \mathbb{N}$, there exists a randomized circuit R^D that solves $\Pi_{c,\alpha}$ on inputs of length $N = 2^n$. We take α' of Lemma 1.18 (depending on c, α), and choose $n \in \mathbb{N}$ so that $m = 2^{\alpha' n}$.[9]

[7]That is, $\Pr_z\left[D(G^f(z)) = 1\right] - \Pr_w\left[D(w) = 1\right] \ge 1/4$; see Section 2 for the definition of PRG.

[8]In the taxonomy of [64], $G^{(\text{-})}$ is *fully black-box* in the sense that the construction from f to G^f also works for every function f.

[9]More precisely, we consider infinitely many m such that D avoids H_m; then, setting $n := (\log m)/\alpha'$, we prove that the reduction R^D solves $\Pi_{c,\alpha}$ on inputs of length n.

The correctness of the reduction R^D can be proved as follows. Assume that f is a YES instance of the promise problem $\Pi_{c,\alpha}$; this means that $\mathrm{Kt}(f) \le \log 2^{cn} = cn$. It follows from the explicitness of G^f (Lemma 1.18) that we can take a large constant c_0 such that $\mathrm{Kt}(G^f(z)) \le c_0\alpha' n = c_0 \log m$ for every seed $z \in \{0,1\}^{O(n)}$. By the definition of the universal hitting set generator H_m, one can observe that $G^f(z) \in \{0,1\}^m$ is in the image of H_m (for infinitely many m); therefore, $G^f(z)$ is rejected by D for every z, and hence the reduction R^D accepts f with probability 1.

Conversely, we prove that any No instance f of $\Pi_{c,\alpha}$ is rejected by the reduction R^D with probability at least $\frac{1}{4}$. Intuitively, this is because of the fact that if f is a hard function, then D cannot distinguish $G^f(\text{-})$ from the uniform distribution. More formally, we prove the contrapositive. Assume that R^D rejects $G^f(z)$ with probability at most $\frac{1}{4}$. This means that

$$\Pr_z[D(G^f(z)) = 1] \le \frac{1}{4}.$$

On the other hand, since D avoids H_m, we have $\Pr_w[D(w) = 1] \ge \frac{1}{2}$. Therefore,

$$\Pr_w[D(w) = 1] - \Pr_z[D(G^f(z)) = 1] \ge \frac{1}{4},$$

which means that $\neg D$ $\frac{1}{4}$-distinguishes G^f from the uniform distribution. By using the black-box security property of G^f (Lemma 1.18), we obtain that $f \in \mathsf{SIZE}^D(2^{(\alpha-\alpha')n})$. Since D is a linear-size circuit (i. e., of size $m = 2^{\alpha' n}$), we conclude that $f \in \mathsf{SIZE}(2^{\alpha n})$, which means that f is not a No instance of the promise problem $\Pi_{c,\alpha}$. Note here that we rely on the efficiency of D, which makes the security proof of the HSG H non-black-box.

To summarize, the promise problem $\Pi_{c,\alpha}$ can be solved by the randomized circuit R^D of size $N^{\beta-2} + 2^{\alpha' n} \le N^{\beta-1.5}$, where the size bound follows from the constructivity of G^f (Lemma 1.18). By using the standard trick of Adleman [1], the randomized circuit R^D of size $N^{\beta-1.5}$ can be converted to a deterministic circuit of size $O(N^{\beta-0.5})$ (by amplifying the success probability and fixing a good random coin flip). We conclude that the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem can be solved by a circuit of size N^β.

(Item 2 $\Rightarrow$ 1) Conversely, if there exists a hitting set generator H secure against linear-size circuits, then the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem cannot be solved by circuits of size $N^{\beta'}$ for any constant β'. Indeed, we can prove the contrapositive (roughly) as follows. Given a circuit D that solves the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem, we claim that the circuit $\neg D$ avoids H. On one hand, any string x in the image of H satisfies that $\mathrm{Kt}(x) = O(\log |x|)$; therefore, x is a YES instance of the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem, and hence rejected by $\neg D$. On the other hand, a string x chosen uniformly at random has circuit complexity at least $2^{\alpha n}$ with high probability for any constant $\alpha < 1$; therefore, $\neg D$ accepts most inputs. (See Proposition 4.22 for a formal proof.) □

Note that the proof above shows a generic connection between a black-box pseudorandom generator construction and a "non-disjoint" promise problem. The efficiency of the security reduction depends on the choice of a black-box PRG construction. For example, Theorem 1.11 is proved by giving an efficient implementation of the black-box PRG construction $G^{(\text{-})}$ of [44] such that $G^f(z)$ is computable by one layer of XOR gates for every fixed seed z. In the rest of this paper, we present some of the instantiations of the proof ideas above based on several specific constructions of PRGs; however, we emphasize that our reductions are not limited to those specific instantiations, and new black-box PRG constructions can lead to a more efficient reduction and a "non-disjoint" promise problem that is easy to analyze.

1.5 Perspective: Meta-computational view of complexity theory

More broadly, we propose to view complexity theory from a meta-computational perspective.

In order to explain the view, it is helpful to regard an algorithm that tries to solve MCLPs as a malicious prover that tries to falsify a circuit lower bound. To be more specific, consider the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem. As we explained earlier, the existence of any algorithm (of any complexity) that solves the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem implies that $\mathsf{E} \subseteq \mathsf{SIZE}(2^{o(n)})$, and moreover the converse direction is also true (for non-uniform algorithms). In this sense, any algorithm that solves an MCLP can be regarded as an adversary that falsifies circuit lower bounds such as $\mathsf{E} \not\subset \mathsf{SIZE}(2^{o(n)})$.

Complexity theory can be regarded as a game between us (i. e., complexity theorists, who try to prove circuit lower bounds) and provers (i. e., algorithms that solve MCLPs). We lose the game if some prover can solve MCLPs (and hence circuit lower bounds fail). We win the game if we find an explicit function whose circuit complexity is high. This is equivalent to finding a witness for the non-disjointness of the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem, and thus it is equivalent to showing that there exists no prover that can solve the MCLP. In other words, prior to our work, we implicitly tried to fight against *every prover without any restriction on efficiency*.

What we showed in this work is that we do not have to fight against every non-efficient prover. Instead, in order to obtain a circuit lower bound (which is implied by the existence of a hitting set generator), it suffices to show that no *efficient* algorithms such as nearly linear-size $\mathsf{AC}^0 \circ \mathsf{XOR}$ circuits can find the difference between explicit functions and hard functions. In principle, it should be easier to prove a nearly linear circuit size lower bound for some problem when we believe that the problem does not admit any algorithm (because of the non-disjointness). While we have not found any existing method for proving such a lower bound that is sufficient for breakthrough results, we believe that this is simply because of the fact that MCLPs were not investigated explicitly. We leave as an important open question to develop a proof technique to analyze MCLPs.

2 Preliminaries

2.1 Notation

For a Boolean function $f : \{0,1\}^n \to \{0,1\}$, we denote by $\mathtt{tt}(f)$ the truth table of f, i. e., the concatenation of $f(x)$ for all $x \in \{0,1\}^n$ in the lexicographical order. We often identify $f : \{0,1\}^n \to \{0,1\}$ with $\mathtt{tt}(f) \in \{0,1\}^{2^n}$. We denote by $\mathrm{size}(f)$ the minimum circuit size for computing the Boolean function $f : \{0,1\}^n \to \{0,1\}$. For a parameter δ, we denote by $\widetilde{\mathrm{size}}(f;\delta)$

the minimum size of a circuit C such that $f(x) = C(x)$ on at least a $(1-\delta)$ fraction of the n-bit inputs x.

For a function $f\colon \{0,1\}^n \to \{0,1\}$ and an integer $k \in \mathbb{N}$, we denote by $f^k\colon (\{0,1\}^n)^k \to \{0,1\}^k$ the direct product of f. We denote by $f^{\oplus k}\colon (\{0,1\}^n)^k \to \{0,1\}$ the function $\oplus_k \circ f^k$, where $\oplus_k$ is the parity function on k-bit inputs.

2.2 Pseudorandomness

Let $G\colon \{0,1\}^d \to \{0,1\}^m$ and $D\colon \{0,1\}^m \to \{0,1\}$ be functions. For an $\epsilon > 0$, we say that D *ϵ-distinguishes* the output distribution of G(-) from the uniform distribution if

$$\Pr_{z\sim\{0,1\}^d}[D(G(z)) = 1] - \Pr_{w\sim\{0,1\}^m}[D(w) = 1] \geq \epsilon$$

In this case, we refer D as an *ϵ-distinguisher* for G. Conversely, G is said to *ϵ-fool* D if D is not an ϵ-distinguisher for G. Similarly, we say that D *ϵ-avoids* G if $\Pr_{w\sim\{0,1\}^m}[D(w) = 1] \geq \epsilon$ and $D(G(z)) = 0$ for every $z \in \{0,1\}^d$. By default, we assume that $\epsilon := \frac{1}{2}$.

For a circuit class $\mathfrak{C}$ and functions $s\colon \mathbb{N} \to \mathbb{N}$ and $\epsilon\colon \mathbb{N} \to [0,1]$, a family of functions $G = \{G : \{0,1\}^{s(n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ is said to be a *pseudorandom generator* that ϵ-fools $\mathfrak{C}$ if, for all large $n \in \mathbb{N}$ and any circuit $C \in \mathfrak{C}$ of n inputs, G $\epsilon(n)$-fools C. We say that G is a *hitting set generator* secure against $\mathfrak{C}$ if for all large $n \in \mathbb{N}$, there is no circuit $D \in \mathfrak{C}$ on n inputs that avoids G_n.

2.3 Circuits

We measure circuit size by the number of gates (except for the input gates). For a circuit type $\mathfrak{C}$ and $s\colon \mathbb{N} \to \mathbb{N}$ and $\delta \in [0,1]$, we denote by $\widetilde{\mathfrak{C}}(s;\delta)$ the class of functions $f\colon \{0,1\}^n \to \{0,1\}$ such that there exists a circuit of size $s(n)$ such that $\Pr_{x\sim\{0,1\}^n}[f(x) = C(x)] \geq 1-\delta$. We define $\mathfrak{C}(s) := \widetilde{\mathfrak{C}}(s;0)$. For the standard circuit class, we use the notation $\widetilde{\mathsf{SIZE}}(s;\delta)$ and $\mathsf{SIZE}(s)$.

2.4 Time-bounded kolmogorov complexity

We fix an efficient universal Turing machine U. Time-bounded Kolmogorov complexity is defined as follows.

Definition 2.1 (Time-bounded Kolmogorov Complexity)**.** The time-t A-oracle Kolmogorov complexity of a string $x \in \{0,1\}^*$ is defined as

$$\mathrm{K}^{t,A}(x) := \min\{\, |d| \mid U^A \text{ outputs } x \text{ in } t \text{ steps on input } d \in \{0,1\}^* \,\},$$

where A is an oracle and $t \in \mathbb{N}$.

3 Meta-computational view of cryptographic PRG constructions

In this section, we apply the meta-computational principle to cryptographic pseudorandom generator constructions. The worst-case-to-average-case reduction for $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}} \text{ vs } \mathrm{K})$ is given in Section 3.1. We view the pseudorandom generator construction of Håstad, Impagliazzo, Levin, and Luby [30] from a meta-computational perspective in Section 3.2.

3.1 $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}} \text{ vs } \mathrm{K})$

We present a proof of the following result.

Reminder of Theorem 1.4. Let A be any oracle. If $\mathsf{DistNP}^A \subseteq \mathsf{AvgP}$, then $\mathrm{Gap}(\mathrm{K}^A \text{ vs } \mathrm{K}) \in \mathsf{P}$.

At the core of the proof of Theorem 1.4 is to use a black-box PRG construction whose advice complexity is small. Following [35], we observe that a k-wise direct product generator, which is one of the simplest constructions of pseudorandom generators, has small advice complexity.

Theorem 3.1 (k-wise direct product generator [35]). *For any parameters $n, k \in \mathbb{N}$ and $\epsilon > 0$ with $k \leq 2n$, there exists a "black-box pseudorandom generator construction" $(\mathrm{DP}_k, A^{(\text{-})}, R^{(\text{-})})$ satisfying the following.*

- $\mathrm{DP}_k \colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{d+k}$ *is called a* pseudorandom generator function *and computable in time* $\mathsf{poly}(n/\epsilon)$. *(Intuitively,* DP_k *takes the n-bit truth table of a candidate "hard" function as well as a d-bit seed and outputs a pseudorandom sequence of length $d + k$.)*
- $A^D \colon \{0,1\}^n \times \{0,1\}^r \to \{0,1\}^a$ *is called an* advice function *and computable in time* $\mathsf{poly}(n/\epsilon)$ *given oracle access to $D \colon \{0,1\}^{d+k} \to \{0,1\}$.*
- $R^D \colon \{0,1\}^a \times \{0,1\}^r \to \{0,1\}^n$ *is called a* reconstruction procedure *and computable in time* $\mathsf{poly}(n/\epsilon)$ *given oracle access to $D \colon \{0,1\}^{d+k} \to \{0,1\}$.*
- *The seed length d is at most $\mathsf{poly}(n/\epsilon)$, the advice complexity a is at most $k + O(\log(k/\epsilon))$, and the randomness complexity r is at most $\mathsf{poly}(n/\epsilon)$.*
- *For any string x and any function D that ϵ-distinguishes the output distribution of $\mathrm{DP}_k(x, \text{-})$ from the uniform distribution, it holds that*

$$\Pr_{w \sim \{0,1\}^r}[R^D(A^D(x, w), w) = x] \geq 3/4.$$

For completeness, we present a proof of Theorem 3.1. We make use of the following list-decodable error-correcting code, which can be constructed by concatenating a Reed-Solomon code with an Hadamard code.

Lemma 3.2 ([67]; see, e.g., [74]). *For any parameters $n \in \mathbb{N}$ and $\epsilon > 0$, there exists a function $\mathrm{Enc}_{n,\epsilon} \colon \{0,1\}^n \to \{0,1\}^{\widehat{n}}$ such that*

- $\widehat{n} = 2^{\ell}$ *for some integer $\ell \in \mathbb{N}$ and $\widehat{n} \leq \mathsf{poly}(n/\epsilon)$,*

- $\mathrm{Enc}_{n,\epsilon}$ *is computable in time* $\mathsf{poly}(n/\epsilon)$*, and*
- *given* $y \in \{0,1\}^{\widehat{n}}$*, one can find a list of size* $\mathsf{poly}(1/\epsilon)$ *that contains all the strings* $x \in \{0,1\}^n$ *such that* y *and* $\mathrm{Enc}_{n,\epsilon}(x)$ *agree on at least a* $(1/2+\epsilon)$*-fraction of coordinates.*

Proof of Theorem 3.1. We describe the pseudorandom generator function

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

which we call a *k-wise direct product generator*. Let $\mathrm{Enc}_{n,\delta}$ denote the error-correcting code of Lemma 3.2, where $\delta := \epsilon/2k$, and let $\widehat{x} \colon \{0,1\}^\ell \to \{0,1\}$ be the function specified by the truth table $\mathrm{Enc}_{n,\delta}(x) \in \{0,1\}^{2^\ell}$, where $\ell = O(\log(n/\epsilon))$. The k-wise direct product generator is defined as

$$\mathrm{DP}_k(x, \bar{z}) := (z^1, \cdots, z^k, \widehat{x}(z^1), \cdots, \widehat{x}(z^k))$$

for any $\bar{z} = (z^1, \cdots, z^k) \in \left(\{0,1\}^\ell\right)^k$. The seed length of DP_k is $d := \ell k = O(k \log(n/\epsilon))$.

We first present a reconstruction procedure R_0 with small success probability; then we will amplify the success probability to 3/4.[10]

Claim 3.3. *There exist an advice function* $A_0 \colon \{0,1\}^n \times \{0,1\}^{r_0} \to \{0,1\}^k$ *and a reconstruction procedure* $R_0^D \colon \{0,1\}^{a_0} \times \{0,1\}^{r_0} \to \{0,1\}^n$ *such that*

$$\Pr_w[R_0^D(A_0(x,w), w) = x] \geq \frac{\epsilon}{2k} \cdot \frac{1}{L},$$

where $r_0 = O(d)$ *and* $L = \mathsf{poly}(k/\epsilon)$ *denotes the list size of Lemma 3.2.*

Proof. Claim 3.3 can be proved by using a standard hybrid argument (as in [56, 74]). Assume that $D \colon \{0,1\}^{d+k} \to \{0,1\}$ satisfies

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

For any $i \in \{0, \cdots, k\}$, define the ith hybrid distribution H_i as the distribution of

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

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

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

[10] We amplify the success probability so that the proof of Theorem 1.4 is clearer, although a proof can be given without the amplification.

By an averaging argument, we obtain

$$\Pr_{\substack{i\sim[k],b \\ z^1,\cdots,z^{i-1},z^{i+1},\cdots,z^k}}\left[\mathop{\mathbb{E}}_{z^i\sim\{0,1\}^\ell}[D(H_i)-D(H_{i-1})]\geq\frac{\epsilon}{2k}\right]\geq\frac{\epsilon}{2k}. \tag{3.1}$$

Define the advice function A_0 as $A_0(x,w):=(\widehat{x}(z^1),\cdots,\widehat{x}(z^{i-1}),b_i,\cdots,b_k)\in\{0,1\}^k$, where w is a coin flip sequence that contains the random choice of $i\sim[k]$, $z^1,\cdots,z^{i-1}\sim\{0,1\}^\ell$, and $b\sim\{0,1\}^k$. By a standard calculation (see, e. g., [74, Proposition 7.16]), it follows from Eq. (3.1) that

$$\Pr_{z^i\sim\{0,1\}^\ell}\left[D(\bar{z},A_0(x,w))\oplus 1\oplus b_i=\widehat{x}(z^i)\right]\geq\frac{1}{2}+\frac{\epsilon}{2k} \tag{3.2}$$

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

Now we describe the reconstruction procedure $R_0^D(\alpha,w)$. Given an advice string $\alpha\in\{0,1\}^k$ and a coin flip w, we regard the random bits w as $i\sim[k]$, $z^1,\cdots,z^{i-1},z^{i+1},\cdots,z^k\sim\{0,1\}^\ell$, $b\sim\{0,1\}^k$, and $j\sim[L]$. Define a string $y\in\{0,1\}^{2^\ell}$ as

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

for every $z^i\in\{0,1\}^\ell$. The reconstruction procedure uses the list-decoding algorithm of Lemma 3.2 to obtain a list of all the 2^ℓ-bit strings that agree with y on a $(1/2+\epsilon/2k)$-fraction of indices, and outputs the jth string in the list.

We claim that $\Pr_w[R_0^D(A_0(x,w),w)=x]\geq\epsilon/2kL$. By Eq. (3.2), with probability at least $\epsilon/2k$ over the random choice of $(i,z^{[k]\setminus\{i\}},b)$, the list-decoding algorithm outputs a list of size L that contains x. With probability at least $1/L$ over the choice of $j\sim[L]$, it holds that x is the jth string in the list, in which case $R_0^D(A_0(x,w),w)=x$. Therefore, we obtain

$$\Pr_w[R_0^D(A_0(x,w),w)=x]\geq\frac{\epsilon}{2k}\cdot\frac{1}{L}.$$

$\diamond$

It remains to amplify the success probability. Intuitively, we repeat picking coin flip sequences $t=O(kL/\epsilon)$ times, and provide the successful index $i\in[t]$ as an advice string. Details follow.

The advice function $A^D\colon\{0,1\}^n\times(\{0,1\}^{r_0})^t\to\{0,1\}^{k+O(\log t)}$ takes an n-bit string x and coin flip sequences $(w_1,\cdots,w_t)\in(\{0,1\}^{r_0})^t$ and outputs $(A_i(x,w_i),i)\in\{0,1\}^k\times[t]$, where i is the first index $i\in[t]$ such that $R_0^D(A_0(x,w_i),w_i)=x$; if there is no such i, the output of A^D is arbitrary. Given an advice string $(\alpha,i)\in\{0,1\}^k\times[t]$ and coin flip sequences $\bar{w}=(w_1,\cdots,w_t)\in(\{0,1\}^{r_0})^t$, we define $R^D((\alpha,i),\bar{w})=R_0^D(\alpha,w_i)$. The failure probability is

$$\Pr_{\bar{w}}\left[R^D(A^D(x,\bar{w}),\bar{w})\neq x\right]=\Pr_{\bar{w}}\left[\forall i\in[t],R_0^D(A_0(x,w_i),w_i)\neq x\right]\leq(1-\epsilon/2kL)^t\leq 1/4,$$

where the last inequality holds for a sufficiently large $t=O(kL/\epsilon)$. $\square$

One of important ingredients of the proof of Theorem 1.4 is a pseudorandom generator constructed by Buhrman, Fortnow, and Pavan [10].

Lemma 3.4 (Buhrman, Fortnow, and Pavan [10]). *If* DistNP $\subseteq$ AvgP*, then there exist a constant* c *and a pseudorandom generator* $G = \{G_n : \{0,1\}^{c \log n} \to \{0,1\}^n\}_{n \in \mathbb{N}}$ *that* $(1/n)$*-fools size-*n *circuits.*

The direct product generator construction (Theorem 3.1) achieves small advice complexity a; however, a disadvantage is that the randomness complexity r is large. The pseudorandom generator of Lemma 3.4 enables us to reduce the randomness complexity effectively.

Another ingredient is the fact that $\mathsf{DistNP}^A \subseteq \mathsf{AvgP}$ implies that a dense subset of A-oracle time-bounded Kolmogorov-random strings can be rejected in polynomial time.

Lemma 3.5 ([32]). *Assume that* $\mathsf{DistNP}^A \subseteq \mathsf{AvgP}$*. Then, there exists a polynomial-time algorithm* M *such that*

1. $M(x, 1^t) = 1$ *for every* x *such that* $\mathrm{K}^{t,A}(x) < |x| - 1$*, and*

2. $\Pr_{x \sim \{0,1\}^n}\left[M(x, 1^t) = 0\right] \geq \frac{1}{4}$ *for every* $n \in \mathbb{N}$ *and every* $t \in \mathbb{N}$*.*

Proof Sketch. We consider a distributional problem $(L, \mathcal{D})$ defined as follows. The language $L \in \mathsf{NP}^A$ is the problem of deciding whether $\mathrm{K}^{t,A}(x) < |x| - 1$ on input $(x, 1^t)$. A distribution $\mathcal{D}_m$ picks $t \sim [m]$ and $x \sim \{0,1\}^{m-t}$ and outputs $(x, 1^t)$. Since the fraction of YES instances in L is at most $\frac{1}{2}$ for any fixed t, any errorless heuristic algorithm that solves $(L, \mathcal{D})$ must reject a $\frac{1}{4}$-fraction of inputs. Details can be found in [32]. $\square$

Now we are ready to prove Theorem 1.4.

Proof of Theorem 1.4. Under the assumption that $\mathsf{DistNP}^A \subseteq \mathsf{AvgP}$, we have the secure pseudorandom generator $G = \{G_m : \{0,1\}^{c_0 \log m} \to \{0,1\}^m\}_{m \in \mathbb{N}}$ of Lemma 3.4. In particular, by replacing random bits used by randomized algorithms with the output of G, we obtain Promise-BPP = Promise-P; thus it suffices to present a randomized polynomial-time algorithm M_1 for computing $\mathrm{Gap}_\tau(\mathrm{K}^A \text{ vs } \mathrm{K})$ for some polynomial τ.

Fix any input $(x, 1^s, 1^t)$, where $x \in \{0,1\}^*$ is a string of length $n \in \mathbb{N}$ and $s, t \in \mathbb{N}$. Take the k-wise direct product generator $\mathrm{DP}_k(x, \text{-})\colon \{0,1\}^d \to \{0,1\}^{d+k}$ of Theorem 3.1, where k is some parameter chosen later and $\epsilon := \frac{1}{8}$. Let M_0 be the polynomial-time algorithm M of Lemma 3.5. Let τ_0 be some polynomial chosen later.

The randomized algorithm M_1 operates as follows. On input $(x, 1^s, 1^t)$, M_1 samples a string $z \sim \{0,1\}^d$ uniformly at random. Then, M_1 simulates M_0 on input $(\mathrm{DP}_k(x, z), 1^{t'})$, where $t' := \tau_0(n, t)$, and accepts if and only if M_0 accepts. (That is, $M_1(x, 1^s, 1^t)$ is defined to be $M_0(\mathrm{DP}_k(x, z), 1^{t'})$ for a random $z \sim \{0,1\}^d$.)

We claim the correctness of the algorithm M_1 below.

Claim 3.6.

1. *If* $\mathrm{K}^{t,A}(x) \leq s$*, then* $M_1(x, 1^s, 1^t)$ *accepts with probability* 1*.*

2. *If* $\mathrm{K}^{\tau(n,t)}(x) > s + \log \tau(n, t)$*, then* $M_1(x, 1^s, 1^t)$ *rejects with probability at least* $\frac{1}{8}$*.*

We claim the first item. Fix any $z \in \{0,1\}^d$. Since the output $\mathrm{DP}_k(x,z)$ of the direct product generator can be described by $n, k \in \mathbb{N}$, the seed $z \in \{0,1\}^d$ and the program for describing x of size $\mathrm{K}^{t,A}(x)$ in time $t' = \tau_0(n,t)$, where τ_0 is some polynomial, it holds that

$$\mathrm{K}^{t',A}(\mathrm{DP}_k(x,z)) \le d + \mathrm{K}^{t,A}(x) + c_1 \log n, \tag{3.3}$$

for some constant c_1. We set $k := s + c_1 \log n + 2$. Note that under the assumption that $\mathrm{K}^{t,A}(x) \le s$, Eq. (3.3) is less than $d + k - 1$. Therefore, by Lemma 3.5, $M_0(\mathrm{DP}_k(x,z)) = 1$ for every $z \in \{0,1\}^d$; thus M_1 accepts.

Next, we claim the second item, by proving its contrapositive. Assume that $M_1(x, 1^s, 1^t)$ rejects with probability less than $\frac{1}{8}$. This means that

$$\Pr_{z \sim \{0,1\}^d}\left[M_0(\mathrm{DP}_k(x,z), 1^t) = 0\right] < \frac{1}{8}.$$

On the other hand, by Lemma 3.5, we also have

$$\Pr_{w \sim \{0,1\}^{d+k}}\left[M_0(w, 1^t) = 0\right] \ge \frac{1}{4}.$$

Therefore,

$$\Pr_{w \sim \{0,1\}^{d+k}}\left[M_0(w, 1^t) = 0\right] - \Pr_{z \sim \{0,1\}^d}\left[M_0(\mathrm{DP}_k(x,z), 1^t) = 0\right] \ge \frac{1}{8},$$

which means that a function D defined as $D(w) := \neg M_0(w, 1^t)$ distinguishes $\mathrm{DP}_k(x, \text{-})$ from the uniform distribution. By the reconstruction property of Theorem 3.1,

$$\Pr_{\rho \sim \{0,1\}^r}\left[R^D(A^D(x,\rho),\rho) = x\right] \ge \frac{3}{4}, \tag{3.4}$$

where the advice complexity a is at most $k + O(\log(k/\epsilon)) = s + O(\log(n/\epsilon))$.[11] Now we derandomize the random choice of ρ of Eq. (3.4) by using the secure pseudorandom generator $G = \{G_m\}_{m \in \mathbb{N}}$. That is, we argue that ρ can be replaced with $G_m(\rho_0)$ for some short seed ρ_0, which enables us to obtain a short description for x. To this end, consider a statistical test $T\colon \{0,1\}^r \to \{0,1\}$ that checks the condition of Eq. (3.4); that is, $T(\rho) = 1$ if and only if $R^D(A^D(x,\rho),\rho) = x$. One can easily observe that T can be implemented by a circuit of size $m := \mathsf{poly}(n,t)$.

Now we replace the random bits $\rho \in \{0,1\}^r$ with the first r bits of the pseudorandom sequence $G_m(\rho_0)$. By Eq. (3.4), we have

$$\Pr_{\rho_0 \sim \{0,1\}^{c_0 \log m}}\left[T(G(\rho_0)) = 1\right] \ge \Pr_{\rho \sim \{0,1\}^r}\left[T(\rho) = 1\right] - o(1) > 0.$$

In particular, there exists a seed $\rho_0 \in \{0,1\}^{c_0 \log m}$ such that $T(G_m(\rho_0)) = 1$.

[11] We may assume without loss of generality that $k := s + O(\log n) \le 2n$, as otherwise $\mathrm{K}^{t,A}(x) \le s$ always holds.

We are ready to present the algorithm for describing x. In order to describe x, it takes as a description $n, t, m \in \mathbb{N}$, the seed $\rho_0 \in \{0,1\}^{c_0 \log m}$, and the advice string $\alpha := A^D(G_m(\rho_0)) \in \{0,1\}^a$. Since $T(G_m(\rho_0)) = 1$, the string x is equal to $R^D(\alpha, G_m(\rho_0))$, which can be computed in time $\tau_1(n,t)$ for some polynomial τ_1. Therefore,

$$\mathrm{K}^{\tau_1(n,t)}(x) \le a + c_0 \log m + O(\log nt) \le s + O(\log nt).$$

In particular, by choosing a polynomial τ large enough, we have

$$\mathrm{K}^{\tau(n,t)}(x) \le \mathrm{K}^{\tau_1(n,t)}(x) \le s + \log \tau(n,t).$$

This completes the proof of Claim 3.6.

Since M_1 is a one-sided-error algorithm, the success probability can be amplified by repeating the computation of M_1 for independent random coin flips. We thus conclude that $\mathrm{Gap}_\tau(\mathrm{K}^A \text{ vs } \mathrm{K})$ is in Promise-BPP = Promise-P. ◻

An important corollary is that NP-hardness of $\mathrm{Gap}(\mathrm{K}^A \text{ vs } \mathrm{K})$ for some $A \in \mathsf{PH}$ is sufficient for proving an equivalence between worst-case and average-case complexity of PH.

Restatement of Corollary 1.6. Assume that $\mathrm{Gap}(\mathrm{K}^A \text{ vs } \mathrm{K})$ is "NP-hard" for some $A \in \mathsf{PH}$ in the sense that

$$\mathsf{NP} \not\subseteq \mathsf{BPP} \quad \Longrightarrow \quad \mathrm{Gap}(\mathrm{K}^A \text{ vs } \mathrm{K}) \notin \mathsf{P}.$$

Then,

$$\mathsf{DistPH} \not\subseteq \mathsf{AvgP} \quad \Longleftrightarrow \quad \mathsf{PH} \neq \mathsf{P}.$$

Proof. It is obvious that $\mathsf{PH} = \mathsf{P}$ implies that $\mathsf{DistPH} \subseteq \mathsf{AvgP}$. We prove the converse direction. Assume that $\mathsf{DistPH} \subseteq \mathsf{AvgP}$. By Theorem 1.4, we obtain $\mathrm{Gap}(\mathrm{K}^A \text{ vs } \mathrm{K}) \in \mathsf{P}$ for any $A \in \mathsf{PH}$. It follows from the contrapositive of the assumption that $\mathsf{NP} \subseteq \mathsf{BPP}$; moreover, by Lemma 3.4, we also have $\mathsf{BPP} = \mathsf{P}$. Therefore, we obtain $\mathsf{NP} = \mathsf{P}$, which is equivalent to $\mathsf{PH} = \mathsf{P}$. ◻

Another corollary is a relationship between time-bounded Kolmogorov complexity and its PH-oracle version, which can be proved by using the disjointness of $\mathrm{Gap}(\mathrm{K}^A \text{ vs } \mathrm{K})$.

Corollary 3.7. *If* $\mathsf{DistPH} \subseteq \mathsf{AvgP}$, *then, for any* $A \in \mathsf{PH}$, *there exists a polynomial* τ *such that*

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

for any $x \in \{0,1\}^*$ *and* $t \in \mathbb{N}$.

Proof. By Theorem 1.4, under the assumption that $\mathsf{DistNP}^A \subseteq \mathsf{DistPH} \subseteq \mathsf{AvgP}$, there exists an algorithm M such that $M(x, 1^s, 1^t) = 1$ for every x such that $\mathrm{K}^{t,A}(x) \le s$ and $M(x, 1^s, 1^t) = 0$ for every x such that $\mathrm{K}^{\tau(|x|,t)}(x) > s + \log \tau(|x|,t)$, for any $s \in \mathbb{N}$ and $t \in \mathbb{N}$. For any $x \in \{0,1\}^*$ and $t \in \mathbb{N}$, define $s := \mathrm{K}^{t,A}(x)$; since $0 \neq M(x, 1^s, 1^t) = 1$, we obtain $\mathrm{K}^{\tau(|x|,t)}(x) \le s + \log \tau(|x|,t)$. ◻

Under the plausible assumption that $\mathsf{E}^{\mathsf{NP}} \neq \mathsf{E}$, we observe that $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}} \text{ vs } \mathrm{K})$ is non-disjoint.

Proposition 3.8. *If $\mathsf{E}^{\mathsf{NP}} \neq \mathsf{E}$, then, for some polynomial τ_0 and any polynomial τ, there are infinitely many strings x such that $\mathrm{K}^{t,\mathsf{SAT}}(x) = O(\log |x|)$ and $\mathrm{K}^{\tau(|x|)}(x) > \log \tau(|x|)$, where $t := \tau_0(|x|)$.*

Proof. By [12], the assumption is equivalent to $\mathsf{E}^{\mathsf{NP}} \not\subset \mathsf{E}/O(n)$. Take a language $L \in \mathsf{E}^{\mathsf{NP}} \setminus \mathsf{E}/O(n)$. For each $n \in \mathbb{N}$, define x_n to be the truth table of length 2^n that encodes the characteristic function of $L \cap \{0,1\}^n$. Since x_n can be described in time $\tau_0(|x_n|) = \mathsf{poly}(|x_n|)$ given $n \in \mathbb{N}$ and oracle access to SAT, we have $\mathrm{K}^{t,\mathsf{SAT}}(x_n) = O(\log |x_n|)$. On the other hand, if $\mathrm{K}^{\tau(|x_n|)}(x_n) \leq \log \tau(|x_n|)$ for all large $n \in \mathbb{N}$, there exists an advice string of length $\log \tau(|x_n|) = O(n)$ that makes it possible to compute $L \cap \{0,1\}^n$ in time $\mathsf{poly}(\tau(|x_n|)) = 2^{O(n)}$, which contradicts $L \notin \mathsf{E}/O(n)$. ◻

Finally, we observe that the complexity of $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}} \text{ vs } \mathrm{K})$ is closely related to the complexity of MINKT.

Proposition 3.9. *For any polynomial τ, the following hold.*

- $\mathrm{Gap}_\tau(\mathrm{K} \text{ vs } \mathrm{K})$ *is reducible to* $\mathrm{Gap}_\tau(\mathrm{K}^{\mathsf{SAT}} \text{ vs } \mathrm{K})$ *via an identity map.*
- *If* $\mathrm{Gap}_\tau(\mathrm{K}^{\mathsf{SAT}} \text{ vs } \mathrm{K})$ *is disjoint, then* $\mathrm{Gap}_\tau(\mathrm{K}^{\mathsf{SAT}} \text{ vs } \mathrm{K})$ *is reducible to* MINKT*; in particular,* $\mathrm{Gap}_\tau(\mathrm{K}^{\mathsf{SAT}} \text{ vs } \mathrm{K}) \in \mathsf{NP}$.

Proof. The first item is obvious because $\mathrm{K}^{t,\mathsf{SAT}}(x) \leq \mathrm{K}^t(x)$ for any $t \in \mathbb{N}$ and $x \in \{0,1\}^*$. For the second item, let $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ denote $\mathrm{Gap}_\tau(\mathrm{K}^{\mathsf{SAT}} \text{ vs } \mathrm{K})$. Since $(\overline{\Pi_{\mathrm{NO}}}, \Pi_{\mathrm{NO}})$ is a problem of checking whether $\mathrm{K}^{\tau(|x|,t)}(x) \leq s + \log \tau(|x|, t)$ given $(x, 1^s, 1^t)$ as input, it is reducible to MINKT. In particular, $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}}) \in \mathsf{NP}$ holds under the assumption that it is disjoint. ◻

3.2 $\mathrm{Gap}(F \text{ vs } F^{-1})$: PRG construction from one-way functions

Håstad, Impagliazzo, Levin, and Luby [30] showed that a cryptographic pseudorandom generator can be constructed from any one-way function. In this section, we view the black-box PRG construction from a meta-computational perspective, which leads us to the promise problem $\mathrm{Gap}(F \text{ vs } F^{-1})$, which is a problem of asking whether a given function f is computable by a small circuit or f is hard to invert by any small circuit. Here we assume that the function $f\colon \{0,1\}^n \to \{0,1\}^n$ is given as oracle, and we focus on an algorithm that runs in time $\mathsf{poly}(n)$. In other words, we consider a sublinear-time algorithm that is given random access to the truth table of f.

Definition 3.10. For a function $\tau\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, $\mathrm{Gap}_\tau(F \text{ vs } F^{-1})$ is a promise problem $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ defined as follows. The input consists of a size parameter $s \in \mathbb{N}$, an integer $n \in \mathbb{N}$, and black-box access to a function $f\colon \{0,1\}^n \to \{0,1\}^n$.

- The set Π_{YES} consists of inputs (n, s, f) such that $\mathrm{size}(f) \leq s$.
- The set Π_{NO} consists of inputs (n, s, f) such that, for any oracle circuit C of size $\tau(n, s)$,

$$\Pr_{x \sim \{0,1\}^n}\left[C^f(f(x)) \in f^{-1}(f(x))\right] < \frac{1}{2}.$$

Theorem 3.11. *If* $\mathsf{DistNP} \subseteq \mathsf{AvgP}$*, then there exist a polynomial* τ *and a* coRP*-type randomized algorithm* M *that solves* $\mathrm{Gap}_\tau(F \text{ vs } F^{-1}) = (\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ *on input* (n, s) *in time* $\mathsf{poly}(n, s)$*. That is,* M *is a randomized oracle algorithm such that*

1. $\Pr_M[M^f(n, s) = 1] = 1$ *for every* $(n, s, f) \in \Pi_{\mathrm{YES}}$*,*
2. $\Pr_M[M^f(n, s) = 0] \geq \frac{1}{2}$ *for every* $(n, s, f) \in \Pi_{\mathrm{NO}}$*, and*
3. $M^f(n, s)$ *runs in time* $\mathsf{poly}(n, s)$*.*

For the proof, we make use of the following black-box construction of a pseudorandom generator based on any one-way function.

Lemma 3.12 (A black-box PRG Construction from Any OWF [30])**.** *There exists a polynomial* $d = d(n)$ *such that, for a parameter* $m \in \mathbb{N}$*, there exists a polynomial-time oracle algorithm* $G_m^{(\cdot)} \colon \{0,1\}^{d(n)} \to \{0,1\}^m$ *that takes a function* $f \colon \{0,1\}^n \to \{0,1\}^n$ *and there exists an oracle algorithm* R *such that, for any function* $D \colon \{0,1\}^m \to \{0,1\}$*, if* $m > d$ *and*

$$\Pr_{z \sim \{0,1\}^n}\left[D(G^f(z)) = 1\right] - \Pr_{w \sim \{0,1\}^m}[D(w) = 1] \geq \frac{1}{8},$$

then

$$\Pr_{x,R}[R^{f,D}(f(x)) \in f^{-1}(f(x))] \geq \frac{1}{2}.$$

The running time of $G_m^{(\cdot)}$ *and* R *is at most* $\mathsf{poly}(n, m)$*.*

Proof Sketch. Since a weakly one-way function exists if and only if a strongly one-way function exists ([77]), it suffices to present a reduction that inverts f with probability at least $1/\mathsf{poly}(n, m)$. (To be more specific, we first amplify the hardness of f by taking a direct product $f^t(x_1, \cdots, x_t) := (f(x_1), \cdots, f(x_t))$, where t is some appropriately chosen parameter, and then apply the HILL construction to f^t described below.)

We invoke the pseudorandom generator construction G^f_{HILL} of Håstad, Impagliazzo, Levin, and Luby [30] based on f. They presented a security reduction R such that if there exists a function D that distinguishes G^f_{HILL} from the uniform distribution, then an oracle algorithm $R^{f,D}(f(x))$ can compute an element in $f^{-1}(f(x))$ with probability at least $1/\mathsf{poly}(n, m)$ over the choice of $x \sim \{0,1\}^n$ and the internal randomness of R. ◻

Proof of Theorem 3.11. Let $G^{(\cdot)}$ be the black-box pseudorandom generator construction of Lemma 3.12, and let R be the security reduction of Lemma 3.12.

Under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$, by Lemma 3.5, there exists a polynomial-time algorithm M such that $M(x, 1^t) = 1$ for every x such that $\mathrm{K}^t(x) < |x| - 1$, and

$$\Pr_{x \sim \{0,1\}^n}\left[M(x, 1^t) = 0\right] \geq \frac{1}{4}$$

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

The algorithm M' for computing $\mathrm{Gap}(F \text{ vs } F^{-1})$ is defined as follows. Let $f\colon \{0,1\}^n \to \{0,1\}^n$ and $s \in \mathbb{N}$ be inputs. Define $m := p(n,s)$ for some polynomial p chosen later. Pick a random $z \in \{0,1\}^n$. Then M' accepts if and only if $M(G^f_m(z), 1^t)$ accepts for a sufficiently large $t = \mathsf{poly}(n,s)$.

We claim the correctness of M' below.

Claim 3.13.

1. *M' accepts any f such that* $\mathrm{size}(f) \le s$ *with probability* 1.
2. *M' rejects any f such that* $\Pr_x\left[C^f(f(x)) \in f^{-1}(f(x))\right] < \frac{1}{2}$ *with probability at least* $\frac{1}{8}$.

We claim that any f such that $\mathrm{size}(f) \le s$ is accepted by the algorithm M'. Indeed, since f is computable by some circuit of size s and G^f_m is polynomial-time-computable, the output of the generator $G^f_m(z)$ can be described by using the description of the circuit of size s, the seed z of length $d(n)$, and $n, m \in \mathbb{N}$ in polynomial time; thus, for a sufficiently large $t = \mathsf{poly}(n,s)$,

$$\mathrm{K}^t(G^f_m(z)) \le d(n) + \widetilde{O}(s) + O(\log n).$$

Choosing $m = p(n,s)$ large enough, this is bounded by $m-2$. Thus M' accepts.

Conversely, suppose that the algorithm M' accepts with probability at least $\frac{7}{8}$. This means that

$$\Pr_{z\sim\{0,1\}^n}\left[M(G^f_m(z), 1^t) = 1\right] \ge \frac{7}{8}.$$

On the other hand, by Lemma 3.5, we have

$$\Pr_{w\sim\{0,1\}^m}\left[M(w, 1^t) = 1\right] \le \frac{3}{4}.$$

Therefore, $M(\text{-}, 1^t)$ is a distinguisher for G^f_m. It follows from the property of the security reduction R that

$$\Pr_{x,R}\left[R^{f, M(\text{-},1^t)}(f(x)) \in f^{-1}(f(x))\right] \ge \frac{1}{2}.$$

By fixing the internal randomness of R and simulating the polynomial-time algorithm $R^{f,M(\text{-},1^t)}$ by a polynomial-size circuit C^f, we conclude that f can be inverted by the oracle circuit C^f of size $\mathsf{poly}(n,s) =: \tau(n,s)$. Thus f is not a No instance of $\mathrm{Gap}_\tau(F \text{ vs } F^{-1})$. ◻

Remark 3.14. In fact, the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ of Theorem 3.11 can be weakened to the assumption that there exists a P-natural property useful against $\mathsf{SIZE}(2^{o(n)})$ (which is essentially equivalent to an errorless heuristic algorithm for MCSP [37, 32]). Indeed, as in [2, 63, 4], the pseudorandom function generator construction of [26] can be used to construct a black-box pseudorandom generator G^f based on a one-way function f that satisfies $\mathrm{size}(G^f(z)) \le \mathsf{poly}(|z|, \log|G^f(z)|)$ for any seed $z \in \{0,1\}^d$; such a pseudorandom generator G^f can be distinguished from the uniform distribution by using the natural property.

We now explain that $\mathrm{Gap}(F \text{ vs } F^{-1})$ is conjectured to be non-disjoint. An *auxiliary-input one-way function* (AIOWF) $f = \{f_x : \{0,1\}^{p(|x|)} \to \{0,1\}^{q(|x|)}\}_{x \in \{0,1\}^*}$ is a polynomial-time-computable function such that, for some infinite set I, for any non-uniform polynomial-time algorithm A, $\Pr_y \left[A(x, f_x(y)) \in f_x^{-1}(f_x(y))\right] < 1/n^{\omega(1)}$ for all large $n \in \mathbb{N}$ and any $x \in I$. This is a weaker cryptographic primitive than a one-way function (i. e., the existence of a one-way function implies that of an auxiliary-input one-way function). Ostrovsky [59] showed that non-triviality of SZK implies the existence of an auxiliary-input one-way function. (see also [73]). We observe that the existence of an auxiliary-input one-way function implies the non-disjointness of $\mathrm{Gap}(F \text{ vs } F^{-1})$.

Proposition 3.15. *If there exists an auxiliary-input one-way function $f = \{f_x : \{0,1\}^{p(|x|)} \to \{0,1\}^{q(|x|)}\}_{x \in \{0,1\}^*}$, then, for any polynomial τ, $\mathrm{Gap}_\tau(F \text{ vs } F^{-1})$ is non-disjoint.*

Proof. Take an infinite set $I \subseteq \{0,1\}^*$ that is hard for polynomial-size circuits to invert $\{f_x\}_{x \in I}$. Since f is polynomial-time-computable, $\mathrm{size}(f_x) \le n^c$ for some constant c, where $n = |x|$. We set the size parameter $s := n^c$. On the other hand, by the property of AIOWF, for any circuit A of size $\tau(n, s)$, it holds that

$$\Pr_y \left[A(x, f_x(y)) \in f_x^{-1}(f_x(y))\right] < \frac{1}{2},$$

for a sufficiently large $x \in I$. This means that f_x is a YES and NO instance of $\mathrm{Gap}_\tau(F \text{ vs } F^{-1})$. □

An immediate corollary of Theorem 3.11 and Proposition 3.15 is that the existence of AIOWF implies $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$ (which is already shown in [32]). An interesting open question is to prove "NP-hardness" of $\mathrm{Gap}(F \text{ vs } F^{-1})$, which has the following important consequence:

Corollary 3.16. *If, for any polynomial τ, it is "NP-hard" to solve $\mathrm{Gap}_\tau(F \text{ vs } F^{-1})$ in time $\mathsf{poly}(n, s)$, then the worst-case and average-case complexity of NP is equivalent in the sense that $\mathsf{P} \neq \mathsf{NP}$ iff $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$.*

Proof. The assumption that $\mathrm{Gap}(F \text{ vs } F^{-1})$ is "NP-hard" means that, for any polynomial τ, if there exists a coRP-type algorithm that solves $\mathrm{Gap}_\tau(F \text{ vs } F^{-1})$ on input (n, s) in time $\mathsf{poly}(n, s)$, then $\mathsf{NP} \subseteq \mathsf{BPP}$. If $\mathsf{DistNP} \subseteq \mathsf{AvgP}$, then Theorem 3.11 implies that there exists some polynomial τ such that $\mathrm{Gap}_\tau(F \text{ vs } F^{-1})$ can be solved by a coRP-type algorithm in time $\mathsf{poly}(n, s)$. By the assumption, we obtain $\mathsf{NP} \subseteq \mathsf{BPP} = \mathsf{P}$, where the last equality is from Lemma 3.4. □

4 Meta-computational view of complexity-theoretic PRG constructions

This section provides a meta-computational view of *complexity-theoretic* PRG constructions. The section is organized as follows. In Section 4.1, we give definitions of resource-bounded Kolmogorov complexity and universal hitting set generators. In Section 4.2, we interpret the notion of advice using resource-bounded Kolmogorov complexity. Using the (new) notion of advice, we provide the definition of MCLPs in Section 4.3. In Section 4.4, we present a generic connection from black-box PRG constructions to MCLPs. Section 4.6 applies the connection to the specific PRG construction given by Nisan and Wigderson [56], which is reviewed in Section 4.5. Section 4.7 provides a meta-computational view of hardness amplification theorems, which gives rise to reductions among different MCLPs. In Section 4.8, we apply our principle to space-bounded algorithms.

4.1 Universal hitting set generators and Kolmogorov complexity

We review the notion of Kt-complexity and its space-bounded version, and present definitions of universal hitting set generators. Recall the definition of Levin's resource-bounded Kolmogorov complexity.

Definition 4.1 ([50]). For a string $x \in \{0,1\}^*$, Levin's Kt complexity of x is defined as

$$\mathrm{Kt}(x) := \min\{\, |d| + t \mid U \text{ outputs } x \text{ on input } d \text{ in time } 2^t \,\},$$

where U is an efficient universal Turing machine.

It is possible to enumerate all the strings x such that $\mathrm{Kt}(x) \le s$ in time $2^{O(s)}$. The enumeration algorithm gives rise to a universal time-bounded hitting set generator.

Definition 4.2 (Universal time-bounded hitting set generator). For a function $s\colon \mathbb{N} \to \mathbb{N}$, define a *universal time-bounded hitting set generator* $\mathrm{Ht}^s = \{\mathrm{Ht}^s_n : \{0,1\}^{s(n)+1} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ so that $\mathrm{Ht}^s_n(d01^t)$ is equal to the output of U on input d if $U(d) \in \{0,1\}^n$ and $U(d)$ halts in 2^t time steps, where $d \in \{0,1\}^*$ and $t \in \mathbb{N}$.

The hitting set generator Ht is universal in the sense that, if there exists a secure hitting set generator $G_n\colon \{0,1\}^s \to \{0,1\}^n$ computable in time 2^s, then $\mathrm{Ht}^{O(s)}_n\colon \{0,1\}^{O(s)} \to \{0,1\}^n$ is also secure. Indeed, the image of G_n is a subset of that of $\mathrm{Ht}^{O(s)}_n$ because of the following property.

Proposition 4.3 (Universality of Ht). *For every function $s\colon \mathbb{N} \to \mathbb{N}$ and $n \in \mathbb{N}$, the image of Ht^s_n contains every string $x \in \{0,1\}^n$ such that $\mathrm{Kt}(x) \le s(n)$. Moreover, the image of Ht^s_n can be enumerated in time $\widetilde{O}(2^{s(n)+\log n})$.*

Proof. Consider any string x of length n such that $\mathrm{Kt}(x) \le s(n)$. By the definition of Kt-complexity, there exists a description $d \in \{0,1\}^*$ such that $U(d)$ outputs x in time 2^t for some $t \le s(n)$ and $|d| \le s(n) - t$. By the definition of Ht^s_n, we have $\mathrm{Ht}^s_n(d01^t) = x$. ◻

The notion of KS-complexity was introduced in [2] as a space-bounded analogue of Kt-complexity.

Definition 4.4 ([2]). For a string $x \in \{0,1\}^*$, the KS-complexity of x is defined as

$$\mathrm{KS}(x) := \min\{\, |d| + s \mid U \text{ outputs } x \text{ on input } d \text{ in space } s \,\}.$$

One can modify the definition of the universal time-bounded hitting set generator to a space-bounded version.

Definition 4.5 (Universal space-bounded hitting set generator)**.** For a function $s\colon \mathbb{N} \to \mathbb{N}$, define a *universal space-bounded hitting set generator* $\mathrm{HS}^s = \{\mathrm{HS}^s_n : \{0,1\}^{s(n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ so that $\mathrm{HS}^s_n(d01^t)$ is equal to the output of U on input d if $U(d) \in \{0,1\}^n$ and $U(d)$ uses at most t space, where $d \in \{0,1\}^*$ and $t \in \mathbb{N}$.

Proposition 4.6 (Universality of HS)**.** *For every function $s\colon \mathbb{N} \to \mathbb{N}$ and $n \in \mathbb{N}$, the image of HS^s_n contains every string $x \in \{0,1\}^n$ such that $\mathrm{KS}(x) \le s(n)$. Moreover, HS^s_n can be computed in $O(s(n) + \log n)$ space.*

4.2 Length-wise advice and resource-bounded Kolmogorov complexity

In order to define meta-computational circuit lower-bound problems, we modify the standard notion of advice. Usually, a complexity class with advice such as $\mathsf{E}/O(n)$ is defined as a subset of functions $f\colon \{0,1\}^* \to \{0,1\}$ that are defined on all the strings of any length. Here, for any $n \in \mathbb{N}$, we define "$\mathsf{DTIME}(2^{O(n)})/^n O(n)$" as a subset of functions $f\colon \{0,1\}^n \to \{0,1\}$, where the superscript n in "$/^n$" is appended in order to emphasize that it depends on n.

Definition 4.7 (Length-wise advice)**.** For any integers $t, a, n \in \mathbb{N}$, we denote by $\mathsf{DTIME}(t)/^n a$ the class of functions $f\colon \{0,1\}^n \to \{0,1\}$ such that there exists a Turing machine M whose description length[12] is a and that outputs $f(x)$ on input $x \in \{0,1\}^n$ in time t. Similarly, let $\mathsf{DSPACE}(t)/^n a$ denote the class of functions $f\colon \{0,1\}^n \to \{0,1\}$ such that there exists a Turing machine M whose description length is a and that outputs $f(x)$ on input $x \in \{0,1\}^n$ in *space* t.

This definition is slightly different from the standard notion of complexity classes with advice, but these are essentially the same. In order to clarify the difference, for functions $t, a\colon \mathbb{N} \to \mathbb{N}$, let $\mathsf{DTIME}(t)/^{\mathrm{KL}} a$ denote the complexity class $\mathsf{DTIME}(t)$ with a-bit advice strings in the standard sense of Karp and Lipton [46].[13] That is, a function $f\colon \{0,1\}^* \to \{0,1\}$ is in $\mathsf{DTIME}(t)/^{\mathrm{KL}} a$ if and only if there exists a Turing machine M such that, for any $n \in \mathbb{N}$, there exists an advice string $\alpha_n \in \{0,1\}^{a(n)}$ such that M outputs $f(x)$ on input (x, α_n) in time $t(n)$ for every $x \in \{0,1\}^n$. Then, the length-wise advice "$/^n$" and the Karp–Lipton advice "$/^{\mathrm{KL}}$" are equivalent in the following sense.

Fact 4.8. *For any functions $t, a\colon \mathbb{N} \to \mathbb{N}$ and any family of functions $f = \{f_n\colon \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$ (which is identified with a function $f\colon \{0,1\}^* \to \{0,1\}$), the following are equivalent.*

1. *There exists a constant c such that $f \in \mathsf{DTIME}(t')/^{\mathrm{KL}} a'$, where $t'(n) := t(n)^c + c$ and $a'(n) := c \cdot a(n) + c$.*

2. *There exists a constant c such that, for any $n \in \mathbb{N}$, $f_n \in \mathsf{DTIME}(t(n)^c + c)/^n c \cdot a(n) + c$.*

Proof Sketch. If $f \in \mathsf{DTIME}(t')/^{\mathrm{KL}} a'$, then there exists a machine M that takes an advice string α_n on inputs of length n. For each $n \in \mathbb{N}$, define M_n to be the machine that, on input

[12]The *description length* $|d_M|$ of a machine M can be formally defined by using a universal Turing machine U. Specifically, a description $d_M \in \{0,1\}^*$ of M is a string that satisfies $U(d_M, x) = M(x)$ for every x.

[13]It is also common to use the notation $\mathsf{DTIME}(t(n))/^{\mathrm{KL}} a(n)$, where n is an indeterminate.

x, simulates M on input (x, α_n); the description length of M_n is at most $O(|\alpha_n|)$, and thus $f_n \in \mathsf{DTIME}(t(n)^{O(1)})/^n O(a(n))$. Conversely, if, for any $n \in \mathbb{N}$, there exists a Turing machine M_n whose description length is $O(a(n))$, then a universal Turing machine U witnesses $f \in \mathsf{DTIME}(t')/^{\mathrm{KL}} a'$. ◻

The advice "$/^n$" is equivalent to Kt-complexity up to a constant factor in the following sense.

Fact 4.9. *For any function $t\colon \mathbb{N} \to \mathbb{N}$ such that $t(n) \geq n$ and for any family of functions $f = \{f_n\colon \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$, the following are equivalent.*

1. $f_n \in \mathsf{DTIME}(t(n)^{O(1)})/^n O(\log t(n))$ *for all large $n \in \mathbb{N}$.*
2. $\mathrm{Kt}(f_n) = O(\log t(n))$ *for all large $n \in \mathbb{N}$.*

4.3 Meta-computational circuit lower-bound problems (MCLPs)

We define promise problems of distinguishing the truth table of explicit functions (e. g., computable in $\mathsf{DTIME}(2^{cn})/^n cn$) from the truth table of hard functions (e. g., that cannot be computed in $\mathsf{SIZE}(2^{\epsilon n})$). We call these Meta-computational Circuit Lower-bound Problems (MCLPs).

Definition 4.10 (Meta-computational Circuit Lower-bound Problems; MCLPs)**.** Let $\mathcal{E}, \mathcal{D}$ be families of functions. *The $\mathcal{E}$ vs $\mathcal{D}$ Problem* is defined as the following promise problem $(\Pi_{\textsc{Yes}}, \Pi_{\textsc{No}})$.

$$\Pi_{\textsc{Yes}} := \{\, \mathtt{tt}(f) \mid f \in \mathcal{E} \,\},$$
$$\Pi_{\textsc{No}} := \{\, \mathtt{tt}(f) \mid f \notin \mathcal{D} \,\}.$$

We will mainly consider a non-uniform computational model for computing the $\mathcal{E}$ vs $\mathcal{D}$ Problem; for $N \in \mathbb{N}$, we denote by $(\mathcal{E}$ vs $\mathcal{D})_N$ the problem restricted to the input length of N.

We denote by (E vs $\mathcal{D}$) a family of problems $\{\mathsf{E}_c \text{ vs } \mathcal{D}\}_{c\in\mathbb{N}}$, where $\mathsf{E}_c := \bigcup_{n\in\mathbb{N}} \mathsf{DTIME}(2^{cn})/^n cn$. For a circuit class $\mathfrak{C}$, we say that (E vs $\mathcal{D}$) is solved by a $\mathfrak{C}$-circuit of size $s(N)$ and denote by (E vs $\mathcal{D}$) $\in \mathsf{i.o.}\mathfrak{C}(s(N))$ if, for every constant c, there exists a family of $\mathfrak{C}$-circuits $\{C_N\}_{N\in\mathbb{N}}$ of size $s(N)$ such that C_N solves the promise problem $(\mathsf{E}_c$ vs $\mathcal{D})_N$ for infinitely many N. We also denote by $\left(\mathsf{E} \text{ vs } \mathsf{SIZE}(2^{o(n)})\right)$ a family of problems $\{\mathsf{E}_c \text{ vs } \mathsf{SIZE}(2^{\alpha n})\}_{c\in\mathbb{N},\alpha>0}$.

Similarly, $\left(\mathsf{DSPACE}(n) \text{ vs } \mathsf{SIZE}(2^{o(n)})\right)$ denotes $\{\bigcup_{n\in\mathbb{N}} \mathsf{DSPACE}(cn)/^n cn \text{ vs } \mathsf{SIZE}(2^{\alpha n})\}_{c\in\mathbb{N},\alpha>0}$.

The definition of the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem given here is slightly different from Definition 1.10 given earlier. In Definition 1.10, we defined the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem by using the notion of Kt-complexity; however, these definitions are essentially equivalent in light of Fact 4.9.

We justify the notation of $\left(\mathsf{DSPACE}(n) \text{ vs } \mathsf{SIZE}(2^{o(n)})\right)$ below. One can observe that the open question whether $\mathsf{DSPACE}(n) \not\subseteq \mathsf{i.o.SIZE}(2^{o(n)})$ is closely related to the $\mathsf{DSPACE}(n)$ vs $\mathsf{SIZE}(2^{o(n)})$ Problem.

Proposition 4.11. *The following are equivalent.*

1. $\mathsf{DSPACE}(n) \not\subset \mathsf{i.o.SIZE}(2^{\epsilon n})$ *for some constant* $\epsilon > 0$.
2. *No circuit can solve the* $\mathsf{DSPACE}(n)$ vs $\mathsf{SIZE}(2^{o(n)})$ *Problem for all large* $n \in \mathbb{N}$.
3. *No circuit can solve the* $\mathsf{DSPACE}(n)$ vs $\widetilde{\mathsf{SIZE}}(2^{o(n)}; \frac{1}{2} - 2^{-o(n)})$ *Problem for all large* $n \in \mathbb{N}$.
4. *There exist some constants* $c \in \mathbb{N}, \alpha > 0$ *such that, for all large* $n \in \mathbb{N}$, *there exists a function* $f\colon \{0,1\}^n \to \{0,1\}$ *such that* $f \in \mathsf{DSPACE}(cn)/^n cn$ *and* $f \notin \widetilde{\mathsf{SIZE}}(2^{\alpha n}; \frac{1}{2} - 2^{-\alpha n})$.

Proof. First, observe that Item 3 is equivalent to Item 4. Indeed, Item 4 implies Item 3 by the definition. Conversely, if Item 4 does not hold, the $\mathsf{DSPACE}(n)$ vs $\widetilde{\mathsf{SIZE}}(2^{o(n)}; \frac{1}{2} - 2^{-o(n)})$ Problem is disjoint; thus, there exists an exponential size circuit that computes the problem, which means that 4 does not hold.

Next, we claim that Item 1 implies Item 4. By using a locally-decodable error-correcting code, it can be shown that Item 1 is equivalent to $\mathsf{DSPACE}(n) \not\subset \mathsf{i.o.}\widetilde{\mathsf{SIZE}}(2^{\alpha n}; \frac{1}{2} - 2^{-\alpha n})$ for some constant $\alpha > 0$ (see [68, 48]). Take a problem $L \in \mathsf{DSPACE}(n) \setminus \mathsf{i.o.}\widetilde{\mathsf{SIZE}}(2^{\alpha n}; \frac{1}{2} - 2^{-\alpha n})$. For each $n \in \mathbb{N}$, let $f_n\colon \{0,1\}^n \to \{0,1\}$ be the characteristic function of $L \cap \{0,1\}^n$. Then, there exists a constant c such that, for all large $n \in \mathbb{N}$, $f_n \in \mathsf{DSPACE}(cn)/^n cn$ and $f_n \notin \widetilde{\mathsf{SIZE}}(2^{\alpha n}; \frac{1}{2} - 2^{-\alpha n})$, which completes the proof of Item 4. It is immediate from the definition that Item 4 implies Item 2.

We claim that Item 2 implies Item 1. Let $f = \{f_n : \{0,1\}^n \to \{0,1\}\}_{n \in \mathbb{N}}$ be a family of functions such that $f_n \in \mathsf{DSPACE}(cn)/^n cn$ and $f_n \notin \mathsf{SIZE}(2^{\alpha n})$ for all large $n \in \mathbb{N}$. In particular, for all large $n \in \mathbb{N}$, there exists some machine M_n of description length cn that computes $f_n(x)$ on input $x \in \{0,1\}^n$ in space cn. Let $L \subseteq \{0,1\}^*$ be a language such that $(x, M) \in L$ if and only if the description length of a Turing machine M is $c|x|$ and M accepts x in space $c|x|$. It is clear that $L \in \mathsf{DSPACE}(n)$. Moreover, for all large $n \in \mathbb{N}$, the characteristic function of $\{\, x \in \{0,1\}^n \mid (x, M_n) \in L \,\}$ is equal to f_n; thus, it requires circuits of size $2^{\alpha n}$. Therefore, $L \notin \mathsf{i.o.SIZE}(2^{\alpha n/(c+1)})$. ∎

It is also possible to define MCLPs whose non-disjointness characterizes other circuit lower bounds. For example, the $\mathsf{EXP/poly}$ vs $\mathsf{SIZE}(2^{o(n)})$ Problem defined as

$$\left\{ \bigcup_{n \in \mathbb{N}} \mathsf{DTIME}(2^{n^c})/^n n^c \text{ vs } \mathsf{SIZE}(2^{\alpha n}) \right\}_{c \in \mathbb{N}, \alpha > 0}$$

is non-disjoint if and only if $\mathsf{EXP/poly} \not\subset \mathsf{SIZE}(2^{o(n)})$.

4.4 MCLPs from PRG and HSG constructions

We formalize the notion of black-box PRG and HSG construction and then we present a general connection between black-box PRG and HSG constructions and meta-computational circuit lower bound problems.

Definition 4.12 (Black-Box PRG and HSG Construction)**.** Let $G^{(\text{-})}\colon \{0,1\}^d \to \{0,1\}^m$ be an oracle algorithm that expects an oracle of the form $f\colon \{0,1\}^\ell \to \{0,1\}$. Let $\mathfrak{C}$ be a circuit class and $\mathcal{R}^{(\text{-})}$ be an oracle circuit class. The algorithm G is referred to as a *black-box* $\mathfrak{C}$*-pseudorandom generator (resp.* $\mathfrak{C}$*-hitting set generator) construction with* $\mathcal{R}$*-reconstruction* and error parameter ϵ if the following hold.

$\mathfrak{C}$-Construction For any seed $z \in \{0,1\}^d$, there exists a $\mathfrak{C}$-circuit that takes $\mathtt{tt}(f)$ as input and outputs $G^f(z)$.

$\mathcal{R}$-Reconstruction For any function $f\colon \{0,1\}^\ell \to \{0,1\}$ and any function $D\colon \{0,1\}^m \to \{0,1\}$, if D is an ϵ-distinguisher for G^f (resp. D ϵ-avoids G^f), then $f \in \mathcal{R}^D$.

We now present a generic connection between MCLPs and HSG constructions.

Theorem 4.13. *Let $G^{(\text{-})}\colon \{0,1\}^s \to \{0,1\}^m$ be a black-box $\mathfrak{C}$-construction with $\mathcal{R}$-reconstruction that takes a function $f\colon \{0,1\}^n \to \{0,1\}$. Define $N := 2^n$. Let $H\colon \{0,1\}^d \to \{0,1\}^m$ be a function such that $G^f(z) \in \mathrm{Im}(H)$ for every $f \in \mathcal{E}$ and every $z \in \{0,1\}^d$. Let $D\colon \{0,1\}^m \to \{0,1\}$ be any function that ϵ-avoids H. Then, the following hold.*

1. *If G^f is a HSG construction with error parameter ϵ, then $(\mathcal{E}\text{ vs }\mathcal{R}^D)_N \in \mathsf{AND}_{2^s} \circ \mathsf{NOT} \circ D \circ \mathfrak{C}$.*
2. *If G^f is a PRG construction with error parameter $\epsilon/2$, then $(\mathcal{E}\text{ vs }\mathcal{R}^D)_N \in \mathsf{AND}_{O(N/\epsilon)} \circ \mathsf{NOT} \circ D \circ \mathfrak{C}$.*

Proof. Let $G^{(\text{-})}$ be a HSG construction with error parameter ϵ. We first present a randomized circuit for solving $(\mathcal{E}\text{ vs }\mathcal{R}^D)$ on inputs of length N. Let $f\colon \{0,1\}^n \to \{0,1\}$ denote an input. Consider a circuit D_1 such that $D_1(f;z) := D\big(G^f(z)\big)$, where z is an auxiliary input (that will be regarded as non-deterministic bits or random bits). We claim that D_1 can solve $(\mathcal{E}\text{ vs }\mathcal{D})$ co-nondeterministically.

Claim 4.14.

1. *If $f \in \mathcal{E}$, then $D_1(f;z) = 0$ for every $z \in \{0,1\}^s$.*
2. *If $f \notin \mathcal{R}^D$, then $D_1(f;z) = 1$ for some $z \in \{0,1\}^s$.*

Suppose that f is a YES instance of $(\mathcal{E}\text{ vs }\mathcal{D})$, that is, $f \in \mathcal{E}$. By the assumption, we have $G^f(z) \in \mathrm{Im}(H)$. Therefore, by the property of D, we obtain $D_1(f;z) = D\big(G^f(z)\big) = 0$.

Conversely, suppose that $D\big(G^f(z)\big) = 0$ for every $z \in \{0,1\}^s$. This means that D ϵ-avoids G^f. By the reconstruction property of G^f, we obtain $f \in \mathcal{R}^D$. Taking its contrapositive, it follows that if $f \notin \mathcal{R}^D$ then $D_1(f;z) = D\big(G^f(z)\big) = 1$ for some $z \in \{0,1\}^s$. This completes the proof of Claim 4.14.

Now consider a circuit D_2 defined as $D_2(f) := \bigwedge_{z\in\{0,1\}^s} \neg D_1(f;z)$. Then, it follows from Claim 4.14 that the circuit D_2 solves $(\mathcal{E}\text{ vs }\mathcal{R}^D)$ on inputs of length N.

We move on to the case when $G^{(\text{-})}$ is a PRG construction. In this case, we claim that the second item of Claim 4.14 can be strengthened to the following: If $f \notin \mathcal{R}^D$, then $\Pr_{z\sim\{0,1\}^s}[D_1(f;z) = 1] \geq \frac{\epsilon}{2}$.

We prove the contrapositive of this claim. Assume that $\Pr_{z\sim\{0,1\}^s}[D_1(f;z)=1] < \frac{\epsilon}{2}$. Since D ϵ-avoids H, we have $\Pr_{w\sim\{0,1\}^m}[D(w)=1] \geq \epsilon$. Therefore, D $\frac{\epsilon}{2}$-distinguishes $G^f(\text{-})$ from the uniform distribution; by the reconstruction of $G^{(\text{-})}$, we obtain $f \in \mathcal{R}^D$, as desired.

Therefore, $D_1(\text{-};z)$ is a one-sided-error randomized circuit that computes ($\mathcal{E}$ vs $\mathcal{R}^D$) with probability at least $\epsilon/2$. Now define a randomized circuit D_2' such that $D_2'(f) := \bigwedge_{i=1}^{k} \neg D_1(f;z_i)$, where $z_1, \cdots, z_k \sim \{0,1\}^s$ are chosen independently and k is a parameter chosen later. Then, it is easy to see that $D_2'(f)=1$ for any $f\in\mathcal{E}$; on the other hand, for any $f \notin \mathcal{R}^D$, $D_2'(f)=1$ with probability at most $(1-\epsilon/2)^k$, which is less than 2^{-N} by choosing $k=O(N/\epsilon)$ large enough. By using a union bound, one can hardwire random bits in D_2' as in Adleman's trick [1], and obtain a deterministic $\mathsf{AND} \circ \mathsf{NOT} \circ D \circ \mathfrak{C}$ circuit that computes ($\mathcal{E}$ vs $\mathcal{R}^D$). ◻

4.5 The Nisan–Wigderson generator

The pseudorandom generator construction of Nisan and Wigderson [56] is particularly efficient. Indeed, each output bit of the Nisan–Wigderson generator depends on only 1 bit of the truth table of a candidate hard function.

Theorem 4.15 (Nisan–Wigderson Pseudorandom Generator Construction)**.** *For every constant $\gamma > 0$ and any $\ell, m \in \mathbb{N}$, there exists a $\mathfrak{C}$-pseudorandom generator construction $G^{(\text{-})}\colon \{0,1\}^d \to \{0,1\}^m$ with $\mathcal{R}$-reconstruction and error parameter ϵ that takes a function $f\colon \{0,1\}^\ell \to \{0,1\}$ such that*

- $d = O(\ell)$, $m \leq 2^\ell$,
- $\mathfrak{C} = \mathsf{NC}^0_1$, *and*
- $\mathcal{R}^D = \widetilde{D \circ \mathsf{AC}^0_2}(m \cdot 2^{\gamma\ell}; \frac{1}{2} - \frac{\epsilon}{m})$.

Moreover, the output $G^f(z)$ of the PRG can be computed in space $O(\ell)$ given a function f and a seed z as input.

We first recall the construction of the Nisan–Wigderson generator. In order to have a space-efficient algorithm for computing the Nisan–Wigderson generator, we use the following construction of a combinatorial design.

Lemma 4.16 (Klivans and van Melkebeek [48], Viola [76])**.** *For every constant $\gamma > 0$, for any $\ell \in \mathbb{N}$, there exist ℓ-sized subsets $S_1, \cdots, S_{2^\ell}$ of $[d]$ for some $d = O(\ell)$ such that*

1. $|S_i \cap S_j| \leq \gamma \cdot \ell$ *for every distinct* $i, j \in [2^\ell]$, *and*
2. $S_1, \cdots, S_{2^\ell}$ *can be constructed in space* $O(\ell)$.

A nearly disjoint generator ND is defined based on the design.

Definition 4.17. For a string $z \in \{0,1\}^d$ and a subset $S \subseteq [d]$ of indices, z_S denotes the string obtained by concatenating the ith bit z_i of z for every $i \in S$. For ℓ-sized subsets $\mathcal{S} = \{S_1, \cdots, S_m\}$ of $[d]$, define a *nearly disjoint generator* ND: $\{0,1\}^d \to (\{0,1\}^\ell)^m$ as $\mathrm{ND}(z) := (z_{S_1}, \cdots, z_{S_m})$ for every $z \in \{0,1\}^d$.

Using the nearly disjoint generator, the Nisan–Wigderson generator is defined as follows.

Definition 4.18 (Nisan–Wigderson generator [56]). For a Boolean function $f\colon \{0,1\}^\ell \to \{0,1\}$ and for ℓ-sized subsets $\mathcal{S} = \{S_1, \cdots, S_m\}$ of $[d]$, the Nisan–Wigderson generator $\mathrm{NW}^f\colon \{0,1\}^d \to \{0,1\}^m$ is defined as

$$\mathrm{NW}^f(z) := f^m \circ \mathrm{ND}(z) = f(z_{S_1}) \cdots f(z_{S_m}).$$

It was shown in [56] that the pseudorandom generator construction is secure in the following sense.

Lemma 4.19 (Security of the Nisan–Wigderson Generator [56]). *For any functions $T\colon \{0,1\}^m \to \{0,1\}$ and $f\colon \{0,1\}^\ell \to \{0,1\}$, for any $\epsilon > 0$, if*

$$\Pr_{w\sim\{0,1\}^m}[T(w) = 1] - \Pr_{z\sim\{0,1\}^d}\left[T(\mathrm{NW}^f(z)) = 1\right] \geq \epsilon,$$

then there exists a one-query depth-2 oracle circuit $C^T \in T \circ \mathsf{AC}^0_2$ of size $O(m \cdot 2^{\gamma\ell})$ such that

$$\Pr_{x\sim\{0,1\}^\ell}\left[C^T(x) = f(x)\right] \geq \frac{1}{2} + \frac{\epsilon}{m}.$$

Proof of Theorem 4.15. We use the Nisan–Wigderson generator $\mathrm{NW}^{(\text{-})}$ defined in Definition 4.18 (i. e., we define $G^{(\text{-})} := \mathrm{NW}^{(\text{-})}$). $\mathrm{NW}^{(\text{-})}$ is an NC^0_1-PRG construction because each bit of $\mathrm{NW}^f(z)$ for any fixed seed z is equal to one bit of the truth table of f. The $\mathcal{R}$-reconstruction property of $\mathrm{NW}^{(\text{-})}$ follows from Lemma 4.19. □

4.6 Meta-computational view of the Nisan–Wigderson generator

Using the Nisan–Wigderson generator construction, we present a general connection between the existence of hitting set generators and lower bounds for meta-computational circuit lower-bound problems. We consider any family of circuits that is closed under taking projections in the following sense.

Definition 4.20. A class $\mathfrak{C}$ of circuits is said to be *closed under taking projections* if, for any $s \in \mathbb{N}$, for every size-s circuit $C \in \mathfrak{C}$ of n inputs, a circuit C' defined as $C'(x_1, \cdots, x_n) = C(x_{\sigma(1)}, \cdots, x_{\sigma(n)})$ for some function $\sigma\colon [n] \to [n]$ can be simulated by a size-s $\mathfrak{C}$-circuit.

Theorem 4.21. *Let $\mathfrak{C}$ be any circuit class that is closed under taking projections. Suppose that $(\mathsf{E}\ \mathrm{vs}\ \widetilde{\mathfrak{C} \circ \mathsf{AC}^0_2}(2^{\alpha n}; \frac{1}{2} - 2^{-\alpha n})) \notin \mathsf{i.o.AND} \circ \mathsf{NOT} \circ \mathfrak{C}(N^{1+\beta})$ for some constants $\alpha, \beta > 0$. Then, there exists a hitting set generator $G = \{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ computable in time $n^{O(1)}$ and secure against linear-size $\mathfrak{C}$ circuits.*

Proof. We prove the contrapositive. Assume that, for every function $s(m) = O(\log m)$, there exists a linear-size $\mathfrak{C}$ circuit D that avoids the universal hitting set generator Ht^s_m for infinitely many $m \in \mathbb{N}$. Given arbitrary constants $c, \alpha, \beta > 0$, we will choose a small constant $\gamma > 0$, and define $s(m) := c' \log m/\gamma$ for some large constant c'. Then using D that avoids Ht^s_m, we present a $\mathsf{AND} \circ \mathsf{NOT} \circ \mathfrak{C}$-circuit of size $N^{1+\beta}$ that solves the E_c vs $\widetilde{\mathfrak{C} \circ \mathsf{AC}^0_2}(2^{\alpha n}; \frac{1}{2} - 2^{-\alpha n})$ Problem.

Let $f\colon \{0,1\}^n \to \{0,1\}$ denote the input of the MCLP. We use the Nisan–Wigderson generator construction $\mathrm{NW}^f\colon \{0,1\}^d \to \{0,1\}^m$ of Theorem 4.15, where $d = O(n)$, $m = 2^{\gamma n}$, and $\epsilon := \frac{1}{4}$. By Theorem 4.15, this is a black-box NC^0_1-PRG construction with $\widetilde{\mathsf{AC}^0_2}(2^{2\gamma n}; \frac{1}{2} - \frac{1}{4m})$ reconstruction.

In order to apply Theorem 4.13, we claim that $\mathrm{NW}^f(z) \in \mathrm{Im}(\mathrm{Ht}^s_m)$ for every $f \in \mathsf{E}_c$ and every $z \in \{0,1\}^d$. By Theorem 4.15, the output $\mathrm{NW}^f(z)$ can be described by using a seed z and a description for f in time $N^{O(1)}$, and thus

$$\mathrm{Kt}(\mathrm{NW}^f(z)) \le |z| + \mathrm{Kt}(f) + O(\log N) = O(n).$$

In particular, for a large enough constant c', we have

$$\mathrm{Kt}(\mathrm{NW}^f(z)) \le c'n = s(m).$$

By the universality of Ht^s_m (Proposition 4.3) we obtain that $\mathrm{NW}^f(z) \in \mathrm{Im}(\mathrm{Ht}^s_m)$.

By applying Theorem 4.13, we have $(\mathsf{E}_c \text{ vs } \widetilde{D \circ \mathsf{AC}^0_2}(2^{2\gamma n}; \frac{1}{2} - \frac{1}{4m}))_N \in \mathsf{AND}_{O(N)} \circ \mathsf{NOT} \circ D \circ \mathsf{NC}^0_1$. Since D is a $\mathfrak{C}$-circuit of size $m = 2^{\gamma n}$, it follows that $(\mathsf{E}_c \text{ vs } \widetilde{\mathfrak{C} \circ \mathsf{AC}^0_2}(2^{O(\gamma n)}; \frac{1}{2} - \frac{1}{4m}))_N \in \mathsf{AND} \circ \mathsf{NOT} \circ \mathfrak{C}(O(N^{1+\gamma}))$. Note that this holds for infinitely many $N = m^{1/\gamma}$. By choosing γ small enough depending on α, β, the result follows. □

We observe that the converse direction also holds. In particular, for any circuit class $\mathfrak{C} \subseteq \mathsf{P/poly}$ such that $\mathfrak{C}$ is closed under taking projections and $\mathsf{AND} \circ \mathsf{NOT} \circ \mathfrak{C} = \mathfrak{C}$, our reductions in fact establish the equivalence between the existence of a hitting set generator secure against $\mathfrak{C}$ and a $\mathfrak{C}$-lower bound for the E vs $\widetilde{\mathsf{SIZE}}(2^{o(n)}; \frac{1}{2} - 2^{-o(n)})$ Problem.

Proposition 4.22 (Converse of Theorem 4.21)**.** *Let $\mathfrak{C}$ be any circuit complexity class. Suppose that there exists a hitting set generator $G = \{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ computable in time $n^{O(1)}$ and secure against linear-size $\mathfrak{C}$ circuits. Then, for any constant $k \in \mathbb{N}$, for all sufficiently large $N \in \mathbb{N}$, no $\mathsf{NOT} \circ \mathfrak{C}$ circuit of size N^k can solve $\left(\mathsf{E} \text{ vs } \widetilde{\mathsf{SIZE}}(2^{\alpha n}; \frac{1}{2} - 2^{-\alpha n})\right)_N$ for $\alpha := 1/4$ nor $\mathrm{MKtP}[O(\log N), N-1]$ on input length N.*

Proof. We first observe that $\mathrm{MKtP}[O(\log N), N-1]$ is reducible to $(\mathsf{E} \text{ vs } \widetilde{\mathsf{SIZE}}(2^{\alpha n}; \frac{1}{2} - 2^{-\alpha n}))$ via an identity map: Take any function $f \in \widetilde{\mathsf{SIZE}}(2^{\alpha n}; \frac{1}{2} - 2^{-\alpha n})$. Since the truth table of f can be described by a circuit of size N^α and $\log \binom{N}{\le N/2 - N^{1-\alpha}}$ bits of information in time $N^{O(1)}$, the Kt-complexity of $\mathtt{tt}(f)$ is at most

$$\widetilde{O}(N^\alpha) + N \cdot \mathrm{H}_2(1/2 - N^{-\alpha}) + O(\log N) \le N - \Omega(N^{1-2\alpha}),$$

which is much smaller than $N-1$. Therefore, it suffices to prove the result only for $\mathrm{MKtP}[O(\log N), N-1]$.

We prove the contrapositive. Suppose that, for some constant $k \in \mathbb{N}$, for any constant c, for infinitely many $N \in \mathbb{N}$, there exists a $\mathsf{NOT} \circ \mathfrak{C}$ circuit of size N^k that solves the promise problem

MKtP[$c \log N, N-1$]. Consider any family of functions $G = \{G_m : \{0,1\}^{d \log m} \to \{0,1\}^m\}_{m \in \mathbb{N}}$ computable in time m^d for a constant d. Let $c := 4kd$, and take a $\mathsf{NOT} \circ \mathfrak{C}$ circuit $\neg C$ of size N^k that solves MKtP[$c \log N, N-1$] on inputs of length N.

We regard C as a circuit that takes $m := N^k$ input bits by ignoring $m - N$ input bits, and in what follows we claim that the linear-size circuit C avoids G_m. For a string $w \in \{0,1\}^m$, denote by $w \restriction_N$ the first N bits of w.

Let $z \in \{0,1\}^{d \log m}$ be any seed of G_m. Since $G_m(z) \restriction_N$ can be described by $N \in \mathbb{N}$ and $z \in \{0,1\}^{d \log m}$ in time m^d, its Kt complexity is

$$\mathrm{Kt}(G_m(z) \restriction_N) \le \log N + |z| + d \log m + o(\log m) \le 4kd \log N,$$

which means that $G_m(z) \restriction_N$ is a YES instance of MKtP[$c \log N, N-1$] and thus $G_m(z)$ is rejected by C.

Now consider a string $w \sim \{0,1\}^m$ chosen uniformly at random. By a standard counting argument, $\mathrm{Kt}(w \restriction_N) \ge N-1$ with probability at least $\frac{1}{2}$; thus C accepts at least a half of all inputs. Therefore, the function G_m is not secure against C. ◻

Applying Theorem 4.21 to depth-d circuits $\mathfrak{C} := \mathsf{AC}^0_d$, we obtain the following.

Corollary 4.23. *Let d be a constant. Suppose that $(\mathsf{E} \text{ vs } \widetilde{\mathsf{AC}^0_{d+2}}(2^{\alpha n}; \frac{1}{2} - 2^{-\alpha n})) \notin \mathsf{i.o.AC}^0_{d+1}(N^{1+\beta})$ for some constants $\alpha, \beta > 0$. Then, there exists a hitting set generator $G = \{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\}_{n \in \mathbb{N}}$ computable in time $n^{O(1)}$ and secure against linear-size AC^0_d circuits.*

This means that, in order to obtain a nearly optimal hitting set generator for AC^0_d, it suffices to prove that nearly linear-size AC^0_{d+1} circuits cannot distinguish the truth tables of functions in $\mathsf{E}/O(n)$ from the truth tables of functions that cannot be approximated by AC^0_{d+2} circuits.

We present a proof of Theorem 1.12.

Restatement of Theorem 1.12. The following are equivalent.

1. For any constants d, d', there exists a constant $\beta > 0$ such that $(\mathsf{E} \text{ vs } \widetilde{\mathsf{AC}^0_{d'}}(2^{o(n)}; \frac{1}{2} - 2^{-o(n)})) \notin \mathsf{i.o.AC}^0_d(N^{1+\beta})$.

2. For any constant d, there exists a hitting set generator $G = \{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\}_{n \in \mathbb{N}}$ computable in time $n^{O(1)}$ and secure against linear-size AC^0_d circuits.

3. For any constant d, there exist constants $c, \beta > 0$ such that MKtP[$c \log N, N-1$] $\notin \mathsf{i.o.AC}^0_d(N^{1+\beta})$.

4. For any constants d, k, there exists a constant $c > 0$ such that MKtP[$c \log N, N-1$] $\notin \mathsf{i.o.AC}^0_d(N^k)$.

Proof of Theorem 1.12. Item 1 $\Longrightarrow$ Item 2 follows from Corollary 4.23. Item 2 $\Longrightarrow$ Item 4 follows from Proposition 4.22. Item 4 $\Longrightarrow$ Item 3 is obvious. Item 3 $\Longrightarrow$ Item 1 holds because the truth table of any function in $\widetilde{\mathsf{AC}^0}(2^{o(n)}; \frac{1}{2} - 2^{-o(n)})$ has Kt complexity less than $N-1$. ◻

4.7 Hardness amplification and MCLPs

The main advantage of studying MCLPs is that hardness amplification can be naturally regarded as a reduction between two different MCLPs. In this section, we present such reductions.

Impagliazzo and Wigderson [44] gave a derandomized hardness amplification theorem. We use the following generalized version of their result.

Theorem 4.24 (Derandomized hardness amplification)**.** *Let $\gamma > 0$ be an arbitrary constant. There exists a hardness amplification procedure* Amp *that takes a function $f: \{0,1\}^n \to \{0,1\}$ and parameters $\delta, \epsilon > 0$, and returns a Boolean function $\mathrm{Amp}^f_{\epsilon,\delta}: \{0,1\}^{O(n+\log(1/\epsilon))} \to \{0,1\}$ satisfying the following:*

1. *If $\widetilde{\mathrm{size}}(\mathrm{Amp}^f_{\epsilon,\delta}; 1/2-\epsilon) \le s$, then $\widetilde{\mathrm{size}}(f;\delta) \le s \cdot \mathsf{poly}(1/\epsilon, 1/\delta)$.*
2. *For any fixed $y \in \{0,1\}^{O(n+\log(1/\epsilon))}$, there exist strings $v_1, \cdots, v_k \in \{0,1\}^n$ for some $k = O(\log(1/\epsilon)/\delta)$ such that $\mathrm{Amp}^f_{\epsilon,\delta}(y) = f(v_1) \oplus \cdots \oplus f(v_k)$. Moreover, if y is distributed uniformly at random, for each $i \in [k]$, v_i is distributed uniformly at random.*
3. *$\mathrm{Amp}^f_{\epsilon,\delta}(y)$ can be computed in $O(n + \log(1/\delta) + \log(1/\epsilon))$ space, given f, y, ϵ and δ as an input.*

Theorem 4.24 slightly differs from derandomized hardness amplification theorems of [44, 31] in that we are also interested in the case when the hardness parameter δ is $o(1)$, which is not required in a standard application of derandomized hardness amplification theorems. We defer a proof of Theorem 4.24 to Appendix A.

Using Theorem 4.24, we give a reduction among different MCLPs.

Theorem 4.25. *For any constants $\alpha, \delta > 0$, for all sufficiently small $\beta > 0$, there exists a $\mathsf{XOR}_{O(\beta \log N)}$-computable reduction from $(\mathsf{E} \text{ vs } \widetilde{\mathsf{SIZE}}(2^{\alpha n}; \delta))$ to $(\mathsf{E} \text{ vs } \widetilde{\mathsf{SIZE}}(2^{\beta n}; \frac{1}{2} - 2^{-\beta n}))$.*

Proof. The reduction is to simply take the hardness amplification procedure Amp of Theorem 4.24. Specifically, given the truth table of a function $f: \{0,1\}^n \to \{0,1\}$, the reduction maps f to the truth table of $\mathrm{Amp}^f_{\epsilon,\delta}: \{0,1\}^{n'} \to \{0,1\}$, where $\epsilon := 2^{-\beta n}$ and $n' = O(n)$. By the second item of Theorem 4.24, each output of the reduction is computable by a XOR of k bits, where $k = O(\log(1/\epsilon)/\delta) = O(\beta n)$.

We claim the correctness of the reduction. Suppose that $\mathrm{Kt}(f) \le O(\log N)$. Then, by the third item of Theorem 4.24, we obtain that $\mathrm{Kt}(\mathrm{Amp}^f_{\epsilon,\delta}) \le O(\log N)$.

Conversely, suppose that $f \notin \widetilde{\mathsf{SIZE}}(2^{\alpha n}; \delta)$; that is, $\widetilde{\mathrm{size}}(f;\delta) > 2^{\alpha n} > 2^{\beta n} \cdot \mathsf{poly}(1/\epsilon, 1/\delta)$, where the last inequality holds by choosing $\beta > 0$ small enough. By the first item of Theorem 4.24, we obtain that $\widetilde{\mathrm{size}}(\mathrm{Amp}^f_{\epsilon,\delta}; \frac{1}{2} - 2^{-\beta n}) > 2^{\beta n}$, which implies that $\mathrm{Amp}^f_{\epsilon,\delta} \notin \widetilde{\mathsf{SIZE}}(2^{\beta' n'}; \frac{1}{2} - 2^{-\beta' n'})$ for some constant $\beta' > 0$. □

Applying the reduction of Theorem 4.25 to Corollary 4.23, we obtain the following.

Corollary 4.26. *Let $d \in \mathbb{N}, \delta > 0$ be constants. Suppose that $(\mathsf{E} \text{ vs } \widetilde{\mathsf{SIZE}}(2^{\alpha n}; \delta)) \notin \mathsf{i.o.AC}^0_{d+2}(N^{1+\beta})$ for some constants $\alpha, \beta > 0$. Then, there exists a hitting set generator $G = \{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\}_{n \in \mathbb{N}}$ computable in time $n^{O(1)}$ and secure against linear-size AC^0_d circuits.*

Proof. We prove the contrapositive. Under the assumption that no hitting set generator is secure against AC^0_d, it follows from Corollary 4.23 that $(\mathsf{E}\text{ vs }\widetilde{\mathsf{AC}^0_{d+2}}(2^{\beta n}; \frac{1}{2} - 2^{-\beta n})) \in \mathsf{i.o.AC}^0_{d+1}(N^{1+\gamma})$ for any constants $\beta, \gamma > 0$. Our goal is to prove that $(\mathsf{E}\text{ vs }\widetilde{\mathsf{SIZE}}(2^{\alpha n}; \delta)) \in \mathsf{i.o.AC}^0_{d+2}(N^{1+\eta})$ for any constants $\alpha, \eta > 0$.

Fix any constants $\alpha, \eta > 0$. By Theorem 4.25, there exists a $\mathsf{XOR}_{O(\beta n)}$-computable reduction from $(\mathsf{E}\text{ vs }\widetilde{\mathsf{SIZE}}(2^{\alpha n}; \delta))$ to $(\mathsf{E}\text{ vs }\widetilde{\mathsf{AC}^0_{d+2}}(2^{\beta n}; \frac{1}{2} - 2^{-\beta n}))$, for all sufficiently small $\beta > 0$. The latter problem can be solved by an $\mathsf{AC}^0_{d+1}(N^{1+\gamma})$. Thus we obtain an $\mathsf{AC}^0_{d+1}(N^{1+\gamma}) \circ \mathsf{XOR}_{O(\beta n)}$ circuit that computes the former problem. Since $\mathsf{XOR}_{O(\beta n)}$ can be computed by a depth-2 circuit of size $N^{O(\beta)}$, by merging one bottom layer of the AC^0_{d+1} circuit, we obtain a circuit in $\mathsf{AC}^0_{d+2}(N^{1+\gamma+O(\beta)})$ that computes $(\mathsf{E}\text{ vs }\widetilde{\mathsf{SIZE}}(2^{\alpha n}; \delta))$. The result follows by choosing $\beta, \gamma > 0$ small enough depending on $\eta > 0$. ◻

Note that AC^0 circuits are not capable of computing XOR gates of large fan-in. If a computational model can compute XOR gates, it is possible to compute a locally-decodable error-correcting code. Specifically, we provide an efficient reduction from $(\mathsf{E}\text{ vs }\mathsf{SIZE}(2^{o(n)}))$ to $(\mathsf{E}\text{ vs }\widetilde{\mathsf{SIZE}}(2^{o(n)}; \frac{1}{2} - 2^{-o(n)}))$ that is computable by a single layer of XOR gates.

Theorem 4.27 (Reductions by Error-Correcting Codes). *For any constants $\alpha, \gamma > 0$, for all sufficiently small $\beta > 0$, there exists a reduction from* $(\mathsf{E}\text{ vs }\mathsf{SIZE}(2^{\alpha n}))$ *to* $(\mathsf{E}\text{ vs }\widetilde{\mathsf{SIZE}}(2^{\beta n}; \frac{1}{2} - 2^{-\beta n}))$ *that is computable by one layer of $O(N^{1+\gamma})$ XOR gates.*

Lemma 4.28 (see [68, 74, 78]). *For any small constant $\beta > 0$, for all large $n \in \mathbb{N}$, there exists an error-correcting code Enc^f that encodes a function $f : \{0,1\}^n \to \{0,1\}$ as a function $\mathrm{Enc}^f : \{0,1\}^{(1+O(\sqrt{\beta}))n} \to \{0,1\}$ satisfying following:*

1. $\mathrm{size}(f) \le \widetilde{\mathrm{size}}(\mathrm{Enc}^f; \frac{1}{2} - 2^{-\beta n}) \cdot 2^{O(\sqrt{\beta} n)}$.
2. *For any fixed $y \in \{0,1\}^{(1+O(\sqrt{\beta}))n}$, $\mathrm{Enc}^f(y)$ can be computed by an XOR of some bits of the truth table of f.*
3. *Enc^f can be computed in time $2^{O(n)}$.*

Proof Sketch. We use a Reed–Muller code concatenated with a Hadamard code. The crux is that the length of a codeword of can be made small because the query complexity of local-list-decoding algorithms is allowed to be quite large.

Let $f : \{0,1\}^n \to \{0,1\}$. Let $\mathbb{F}_q$ be a finite field, where $q = 2^k$ for some k. Pick $H \subseteq \mathbb{F}_q$, and encode any element of $\{0,1\}^n$ as an element of H^t by taking an injection η from $\{0,1\}^n$ to H^t, where t is some large constant chosen later. Let the size $|H|$ of H be $2^{n/t}$. Any $f : \{0,1\}^n \to \{0,1\}$ can be encoded as a low-degree extension $\widehat{f} : \mathbb{F}_q^t \to \mathbb{F}_q$ such that $\widehat{f}$ and $f \circ \eta^{-1}$ agree on H^t. Specifically, $\widehat{f}$ can be defined as

$$\widehat{f}(x) = \sum_{y \in H^t} f(\eta^{-1}(y)) \prod_{i=1}^{t} \delta_{y_i}(x_i)$$

for $x = (x_1, \ldots, x_t) \in \mathbb{F}_q^t$, where δ_a is a polynomial of degree at most $|H|$ such that $\delta_a(a) = 1$ and $\delta_a(b) = 0$ for every $b \in H \setminus \{a\}$. The total degree of $\widehat{f}$ is at most $d := t|H| = t2^{n/t}$. We will set $q = t2^{n/t+O(\beta n)}$.

Then, each alphabet $\widehat{f}(x)$ in the Reed–Muller code is encoded with a Hadamard code. Namely, $\mathrm{Enc}^f(x, y) := \langle \widehat{f}(x), y \rangle$, where $x \in \mathbb{F}_q^t$, $y \in \mathbb{F}_2^k$ and $\langle -, - \rangle$ denotes the inner product function over $\mathbb{F}_2$. The length of the truth table of Enc^f is at most $q^{t+1} = O(t^{t+1} 2^{(n/t+O(\beta n))(t+1)})$. By choosing $t := 1/\sqrt{\beta}$, this is bounded by $2^{(1+O(\sqrt{\beta}))\cdot n}$. The second item holds because for every fixed x, each coordinate of $\widehat{f}(x) \in \mathbb{F}_q \cong \mathbb{F}_2^k$ can be written as an XOR of some bits of the truth table of f.[14]

To see the first item, we use the local list decoding algorithm of Sudan, Trevisan, and Vadhan [68]. They gave a local list-decoding algorithm for the code Enc^f running in time $\mathsf{poly}(t, q)$ that can handle a $\left(\frac{1}{2} - (d/q)^{\Omega(1)}\right)$-fraction of errors, which is more than $\frac{1}{2} - 2^{-\beta n}$ by choosing $q := t2^{n/t+O(\beta n)}$ large enough. Given a circuit that approximates Enc^f on a $\frac{1}{2} - 2^{-\beta n}$ fraction of inputs, one can apply the local list-decoding algorithm to obtain a circuit that computes f on every input; thus we have $\mathrm{size}(f) \le \widetilde{\mathrm{size}}(\mathrm{Enc}^f; \frac{1}{2} - 2^{-\beta n}) \cdot 2^{O(\sqrt{\beta})n}$. ◻

Proof of Theorem 4.27. We apply the error-correcting code Enc of Theorem 4.27. Specifically, given $f : \{0,1\}^n \to \{0,1\}$ as input, we map f to $\mathrm{Enc}^f : \{0,1\}^{n'} \to \{0,1\}$. Since the length of $\mathtt{tt}(\mathrm{Enc}^f)$ is $2^{n'} = N^{1+O(\sqrt{\beta})}$ and each bit is computable by a XOR of some bits of $\mathtt{tt}(f)$, the reduction is computable by one layer of $O(N^{1+\gamma})$ XOR gates for all sufficiently small $\beta > 0$.

We claim the correctness of the reduction. Suppose that $\mathrm{Kt}(f) = O(\log N)$; then, by the third item of Lemma 4.28, we have $\mathrm{Kt}(\mathrm{Enc}^f) = O(\log N)$.

Now suppose that $f \notin \mathsf{SIZE}(2^{\alpha n})$. By Lemma 4.28, we have

$$2^{\alpha n} < \mathrm{size}(f) \le \widetilde{\mathrm{size}}(\mathrm{Enc}^f; \frac{1}{2} - 2^{-\beta n}) \cdot 2^{O(\sqrt{\beta} n)}.$$

Therefore, we obtain that $\widetilde{\mathrm{size}}(\mathrm{Enc}^f; \frac{1}{2} - 2^{-\beta n}) > 2^{(\alpha - O(\sqrt{\beta}))n} \ge 2^{\beta n}$, where the last inequality holds by choosing $\beta > 0$ small enough. Therefore, $\mathrm{Enc}^f \notin \widetilde{\mathsf{SIZE}}(2^{\beta' n'}; \frac{1}{2} - 2^{-\beta' n'})$ holds for some constant $\beta' > 0$. ◻

Applying the reduction, we obtain the following.

Theorem 4.29. *Let d be a constant. Suppose that $(\mathsf{E}$ vs $\mathsf{SIZE}(2^{\alpha n})) \notin \mathsf{i.o.AC}^0_{d+1} \circ \mathsf{XOR}(N^{1+\beta})$ for some constants $\alpha, \beta > 0$. Then, there exists a hitting set generator $G = \{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\}_{n \in \mathbb{N}}$ computable in time $n^{O(1)}$ and secure against linear-size $\mathsf{AC}^0_d \circ \mathsf{XOR}$ circuits.*

Proof. We prove the contrapositive. Let $\alpha, \beta > 0$ be arbitrary constants. Assuming that no hitting set generator is secure against $\mathsf{AC}^0_d \circ \mathsf{XOR}$, by Theorem 4.21, we have $(\mathsf{E}$ vs $\widetilde{\mathsf{SIZE}}(2^{\alpha_0 n}; \frac{1}{2} - 2^{-\alpha_0 n})) \in \mathsf{i.o.AC}^0_{d+1} \circ \mathsf{XOR}(N^{1+\beta_0})$ for any constants $\alpha_0, \beta_0 > 0$. By Theorem 4.27, $(\mathsf{E}$ vs $\mathsf{SIZE}(2^{\alpha n}))$ is reducible

[14] Note that an addition over $\mathbb{F}_q$ is given by a coordinate-wise addition over $\mathbb{F}_2^k$.

to (E vs $\widetilde{\mathsf{SIZE}}(2^{\alpha_0 n}; \frac{1}{2} - 2^{-\alpha_0 n})$) by one layer of $O(N^{1+\gamma})$ XOR gates for any constant $\gamma > 0$ and any small enough constant $\alpha_0 > 0$. Therefore, we obtain a circuit in $\mathsf{i.o.AC}^0_{d+1} \circ \mathsf{XOR} \circ \mathsf{XOR}((N^{1+\gamma})^{1+\beta_0})$ that computes (E vs $\mathsf{SIZE}(2^{\alpha n})$). By merging the bottom two XOR layers, the circuit can be written as an $\mathsf{i.o.AC}^0_{d+1} \circ \mathsf{XOR}$ circuit of size $N^{1+\gamma+\beta_0+\gamma\beta_0}$.[15] The result follows by choosing positive constants $\gamma, \beta_0 \ll \beta$ small enough. □

We are now ready to complete a proof of Theorem 1.11, which establishes the equivalence between the existence of a hitting set generator secure against $\mathsf{AC}^0 \circ \mathsf{XOR}$ and the $\mathsf{AC}^0 \circ \mathsf{XOR}$ circuit lower bound for the E vs $\mathsf{SIZE}(2^{o(n)})$ Problem.

Restatement of Theorem 1.11. The following are equivalent.

1. For any constant d, there exists a hitting set generator $G = \{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\}_{n \in \mathbb{N}}$ computable in time $n^{O(1)}$ and secure against linear-size $\mathsf{AC}^0_d \circ \mathsf{XOR}$ circuits.

2. For any constant d, for some constant $\beta > 0$, (E vs $\mathsf{SIZE}(2^{o(n)})$) $\notin \mathsf{i.o.AC}^0_d(N^{1+\beta})$.

3. For any constant d, for some constant $\beta > 0$, $\mathrm{MKtP}[O(\log N), N^{o(1)}] \notin \mathsf{i.o.AC}^0_d \circ \mathsf{XOR}(N^{1+\beta})$.

4. For any constants $d, k \in \mathbb{N}$, $\mathrm{MKtP}[O(\log N), N^{o(1)}] \notin \mathsf{i.o.AC}^0_d \circ \mathsf{XOR}(N^k)$.

Proof of Theorem 1.11. The implications from Item 4 to Item 3 and from Item 3 to Item 2 are trivial. The implication from Item 2 to Item 1 immediately follows from Theorem 4.29. The implication from Item 1 to Item 4 is a standard approach for showing a lower bound for MKtP, and follows from Proposition 4.22. □

4.8 KS complexity and read-once branching program

We now turn our attention to KS complexity. This amounts to considering a hitting set generator that is computable in a limited amount of space. Applying our proof ideas to the case of read-once branching programs, we provide a potential approach for resolving $\mathsf{BPL} = \mathsf{L}$.

Restatement of Theorem 1.13. There exists a universal constant $\rho > 0$ satisfying the following. Suppose that, for some constants $\alpha, \beta > 0$, ($\mathsf{DSPACE}(n)$ vs $\widetilde{\mathsf{SIZE}}(2^{\alpha n}; 2^{-\rho\alpha n})$) cannot be computed by a read-once co-nondeterministic branching program of size $N^{1+\beta}$ for all large input length $N \in \mathbb{N}$. Then there exists a hitting set generator $G = \{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\}_{n \in \mathbb{N}}$ computable in $O(\log n)$ space and secure against linear-size read-once branching programs.

Since the class of read-once branching programs is not closed under taking several reductions presented so far, we provide a self-contained proof below.

[15] Recall that we count the number of gates except for input gates.

Proof. We prove the contrapositive of Theorem 1.13 for the universal hitting set generator HS. Assume that, for every function $s(m) = O(\log m)$, there exists a linear-size read-once branching program D that avoids HS^s_m for infinitely many $m \in \mathbb{N}$. Given arbitrary constants $c, \alpha, \beta > 0$, we will choose a small constant $\gamma > 0$, and define $s(m) := c' \log m/\gamma$ for some large constant c'. Then using D that avoids HS^s_m, we present a coRP-type randomized read-once branching program that solves $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}}) := (\mathsf{DSPACE}(cn)/^n cn \text{ vs } \widetilde{\mathsf{SIZE}}(2^{\alpha n}; 2^{-\rho\alpha n}))$ on inputs of length $N = 2^n$, for some sufficiently large $N := m^{1/\gamma}$. Here, a *randomized* branching program means a probability distribution on branching programs.

For simplicity, we first explain a construction of a branching program that may not be read-once. A randomized branching program D_0 is defined as follows. Given an input $f\colon \{0,1\}^n \to \{0,1\}$, set $\epsilon := 1/4m$. Define $\widetilde{f} := \mathrm{Amp}^f_{\epsilon,\delta}\colon \{0,1\}^{O(\log N)} \to \{0,1\}$. Let ℓ denote the input length of $\widetilde{f}$, and let $d = O(\ell) = O(n)$ denote the seed length of the Nisan–Wigderson generator instantiated with ℓ-sized m subsets (Definition 4.18). Pick $z \sim \{0,1\}^d$ and output $\neg D(\mathrm{NW}^{\widetilde{f}}(z))$. This is a randomized branching program because for each fixed z, each bit of $\mathrm{NW}^{\widetilde{f}}(z)$ is equal to $\widetilde{f}(z_S)$ for some subset S, and hence by Item 2 of Theorem 4.24 it is some linear combination of at most k bits of $\mathtt{tt}(f)$, which can be computed by a read-once width-2 branching program of size $O(k)$.[16] Thus $\neg D(\mathrm{NW}^{\widetilde{f}}(z))$ can be implemented as a branching program of size $O(m \cdot k)$, by replacing each node of D by the read-once width-2 branching programs that compute some linear combinations of $\mathtt{tt}(f)$.

We claim the correctness of the randomized branching program D_0.

Claim 4.30.

1. *D_0 accepts every $f \in \Pi_{\mathrm{YES}}$ with probability 1.*
2. *For every $f \in \Pi_{\mathrm{NO}}$, the probability that D_0 rejects f is at least $\frac{1}{4}$.*
3. *The size of D_0 is at most N^β.*

Take any YES instance $f \in \Pi_{\mathrm{YES}}$. We observe that the output $\mathrm{NW}^{\widetilde{f}}(z)$ of the generator has small KS complexity. Indeed,

$$\mathrm{KS}(\mathrm{NW}^{\widetilde{f}}(z)) \le |z| + \mathrm{KS}(\widetilde{f}) + O(\log N) \le \mathrm{KS}(f) + O(\log N),$$

where we used Lemma 4.16 and Item 3 of Theorem 4.24. In particular, for a large enough constant c', we have

$$\mathrm{KS}(f) \le c' \log N = s(m),$$

and thus $D(\mathrm{NW}^{\widetilde{f}}(z)) = 0$ by Proposition 4.6 and the assumption that D avoids HS^s_m. This means that the algorithm D_0 accepts for every choice of z.

[16] Note that, since z is fixed, the branching program does not have to compute z_S from z.

Conversely, suppose that the algorithm D_0 accepts some input f with probability at least $\frac{3}{4}$. We claim that $f \notin \Pi_{\text{No}}$. The assumption means that $\Pr_z[D(\mathrm{NW}^{\widetilde{f}}(z)) = 0] \geq \frac{3}{4}$, which is equivalent to saying that $\Pr_z[D(\mathrm{NW}^{\widetilde{f}}(z)) = 1] \leq \frac{1}{4}$. On the other hand, since D avoids HS^s_m, we have $\Pr_w[D(w) = 1] \geq \frac{1}{2}$. In particular, we obtain

$$\Pr_w[D(w) = 1] - \Pr_z[D(\mathrm{NW}^{\widetilde{f}}(z)) = 1] \geq \frac{1}{4}.$$

By the security proof of the Nisan–Wigderson generator (Lemma 4.19), there exists a one-query oracle circuit C of size $O(m \cdot 2^{\gamma\ell})$ such that

$$\Pr_{y \sim \{0,1\}^\ell}[C^D(y) = \widetilde{f}(y)] \geq \frac{1}{2} + \frac{1}{4m}.$$

Now replacing the oracle gate of C with a circuit that simulates D, we obtain a circuit of size $m^{O(1)} + m \cdot N^{O(\gamma)}$. By the property of $\mathrm{Amp}_{\epsilon,\delta}$ (Item 1 of Theorem 4.24), we obtain another circuit C' of size $m^{O(1)} \cdot N^{O(\gamma)} \cdot (1/\delta)^{O(1)}$ such that

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

Choosing $\delta := N^{-\gamma}$, we obtain that $\widetilde{\mathrm{size}}(f; N^{-\gamma}) \leq N^{O(\gamma)}$, where $\gamma > 0$ is an arbitrary small constant. This completes the proof of the second item of Claim 4.30, by choosing γ small enough so that $O(\gamma) < \alpha$.

Recall that the size of D_0 is $O(m \cdot k)$. Here k is the parameter from Theorem 4.24 and $k = O(\log(1/\epsilon)/\delta) = N^{O(\gamma)}$. Thus the size of D_0 is at most $N^{O(\gamma)} \leq N^\beta$, by choosing γ small enough so that $O(\gamma) \leq \beta$. This completes the proof of Claim 4.30.

Now we modify the construction of D_0 in order to obtain a *read-once* branching program D_1. The branching program $D_0(\mathrm{NW}^{\widetilde{f}}(z))$ may read some x-th bit of the input $\mathtt{tt}(f)$ twice only if there exists a pair of distinct indices (i, j) such that the ith bit of $\mathrm{NW}^{\widetilde{f}}(z)$ and the jth bit of $\mathrm{NW}^{\widetilde{f}}(z)$ are linear combinations of $\mathtt{tt}(f)$ that contain $f(x)$. We say that a coin flip z is *bad* if this happens. To ensure that a coin flip is bad with small probability, we take a pairwise independent generator $G_2 \colon \{0,1\}^{O(\ell)} \to (\{0,1\}^\ell)^m$, and define a modified Nisan–Wigderson generator $\mathrm{NW}'^{\widetilde{f}} := \widetilde{f}^m \circ (\mathrm{ND} \oplus G_2)$, where $\mathrm{ND} \oplus G_2$ denotes the function such that $\mathrm{ND} \oplus G_2(u, v) := \mathrm{ND}(u) \oplus G_2(v)$. Using this modified construction, the read-once branching program D_1 is defined as $\neg D(\mathrm{NW}'^{\widetilde{f}}(z))$ if z is not bad, and otherwise defined as a trivial branching program that outputs 1 always. (Note here that since we deal with a non-uniform computation, one does not need to check the badness of z by using a branching program.) By the definition, it is obvious that D_1 is read-once; hence it remains to claim that D_1 satisfies the promise of $(\Pi_{\text{Yes}}, \Pi_{\text{No}})$.

As in the case of D_0, for $f \in \Pi_{\text{Yes}}$, it can be seen that $D(\mathrm{NW}'^{\widetilde{f}}(z)) = 0$ for every z, and hence D_1 always accepts. (We note that the KS complexity increases by an additive term of the input length of G_2, which is $O(\log N)$.) Conversely, we claim that if D_1 accepts with probability at least $\frac{7}{8}$, then $f \notin \Pi_{\rm No}$. Assume that $\Pr_z[D_1 \text{ accepts}] \geq \frac{7}{8}$. We claim that the probability that z is bad is small: Fix any distinct indices $(i, j) \in [m]^2$. Recall that by Item 2 of Theorem 4.24, there exist functions $h_1, \cdots, h_k$ such that $\widetilde{f}(y) = f(h_1(y)) \oplus \cdots \oplus f(h_k(y))$ for every $y \in \{0,1\}^\ell$, where $k = O(\log(1/\epsilon)/\delta)$. Then, the ith bit of $\mathrm{NW}'^{\widetilde{f}}(u, v)$ is equal to $f(h_1(u_{S_i} \oplus G_2(v)_i)) \oplus \cdots \oplus f(h_k(u_{S_i} \oplus G_2(v)_i))$. Fix any distinct indices $i', j' \in [k]$. Then, we have

$$\Pr_{u,v}[h_{i'}(u_{S_i} \oplus G_2(v)_i) = h_{j'}(u_{S_j} \oplus G_2(v)_j)] = \Pr_{\substack{u \\ a,b \sim \{0,1\}^\ell}}[h_{i'}(u_{S_i} \oplus a) = h_{j'}(u_{S_j} \oplus b)] = \frac{1}{N},$$

where the first inequality holds by the pairwise independence of G_2 and the second inequality holds because $h(a)$ is distributed identically with the uniform distribution for $a \sim \{0,1\}^\ell$. By the union bound over all (i, j) and (i', j'), the probability that z is bad is bounded above by $(km)^2 \cdot 1/N \leq N^{O(\gamma)-1} \leq \frac{1}{8}$ for a sufficiently small $\gamma > 0$ and a large $N \in \mathbb{N}$. Thus we obtain

$$\Pr_z[D(\mathrm{NW}'^{\widetilde{f}}(z)) = 0] \geq \Pr[D_1 \text{ accepts}] - \Pr[z \text{ is bad}] \geq \frac{3}{4}.$$

The rest of a proof of the correctness is essentially the same with the case of D_0, observing that the security proof of the Nisan–Wigderson generator also works for the modified version NW′.

Finally, we convert the randomized read-once branching program D_1 of size N^β into a co-nondeterministic read-once branching program of size $N^{1+\beta}$. This can be done by using the standard Adleman's trick [1]: Specifically, the success probability of D_1 can be amplified to $1 - 2^{-N}$ by taking AND of $O(N)$ independent copies of D_1. By the union bound, there exists a good coin flip sequence such that AND of $O(N)$ copies of D_1 solves $(\Pi_{\rm YES}, \Pi_{\rm No})$ on every input of length N. Hard-wiring such a coin flip sequence and simulating the AND gate by using a co-nondeterministic computation, we obtain a co-nondeterministic read-once branching program of size $O(N^{1+\beta})$. □

5 Non-trivial derandomization and MKtP

In this section, we provide a characterization of non-trivial derandomization for uniform algorithms by a lower bound for MKtP. We start with a formal definition of non-trivial derandomization for uniform algorithms.

Definition 5.1 (Non-trivial derandomization)**.** An algorithm A is said to be a *derandomization algorithm for* $\mathsf{DTIME}(t(n))$ *that runs in time* $s(n)$ if the following hold. The algorithm takes 1^n and a description of a machine M and outputs an n-bit string $A(1^n, M) \in \{0,1\}^n$ in time $s(n)$. For any machine M running in time $t(n)$ on inputs of length $n \in \mathbb{N}$, there exist infinitely many $n \in \mathbb{N}$ such that, if $\Pr_{x \sim \{0,1\}^n}[M(x) = 1] \geq \frac{1}{2}$, then $M(A(1^n, M)) = 1$.

In the following, for a function $s \colon \mathbb{N} \to \mathbb{N}$, we denote by $R^s_{\rm Kt}$ the set $\{\, x \in \{0,1\}^* \mid \mathrm{Kt}(x) \geq s(|x|) \,\}$ of Kt-random strings with threshold s. We say that a set $R \subseteq \{0,1\}^*$ is dense if $\Pr_{x \sim \{0,1\}^n}[x \in R] \geq \frac{1}{2}$ for all large $n \in \mathbb{N}$.

Proposition 5.2. *The following are equivalent for any time-constructible functions $t(n) \geq n$ and $s(n)$.*

1. *There exists a derandomization algorithm for $\mathsf{DTIME}(t(n))$ that runs in time $2^{s(n)+O(\log t(n))}$.*

2. *Any dense subset of $R_{\mathrm{Kt}}^{s(n)+O(\log t(n))}$ cannot be accepted by any $t(n)$-time algorithm.*

Proof. (Item 1 $\Longrightarrow$ Item 2) Let A be a derandomization algorithm for $\mathsf{DTIME}(t(n))$. Since A runs in time $2^{s(n)+O(\log t(n))}$ on input $(1^n, M)$, the Kt-complexity of $A(1^n, M)$ is at most $s(n) + O(\log t(n)) + |M|$.

Let M be any $t(n)$-time algorithm such that $\Pr_{x \sim \{0,1\}^n}[M(x) = 1] \geq \frac{1}{2}$ for all large $n \in \mathbb{N}$. By the property of A, we have $M(A(1^n, M)) = 1$ for infinitely many n. This means that M accepts the string $A(1^n, M)$ that has Kt-complexity at most $s(n) + O(\log t(n))$; therefore, M does not accept a dense subset of Kt-random strings.

(Item 2 $\Longrightarrow$ Item 1) Let A be the algorithm that takes $(1^n, M)$, enumerates all the strings x whose Kt-complexity is at most $s(n)+O(\log t(n))$, and, for each string x with small Kt-complexity, simulates M on input x; if M accepts x in time $t(n)$, output x and halt. The running time of A is clearly at most $2^{s(n)+O(\log t(n))}$. To prove the correctness, assume towards a contradiction that some algorithm M runs in time $t(n)$ and for all large n, $\Pr_{x \sim \{0,1\}^n}[M(x) = 1] \geq \frac{1}{2}$ and $M(A(1^n, M)) = 0$. The latter condition means that A cannot find a string x that is accepted by M; thus, M rejects all the strings x whose Kt-complexity is at most $s(n) + O(\log t(n))$. Therefore, M accepts a dense subset of Kt-random strings, which is a contradiction. ◻

Theorem 5.3. *Let $t, s \colon \mathbb{N} \to \mathbb{N}$ be time-constructible functions such that t is non-decreasing and $\omega(\log^2 n) \leq s(n) \leq n$ and $\mathsf{poly}(n) \leq t(2s(n))$ for all large n, where poly is some universal polynomial.*

For any constant $c > 0$, there exists a constant c' such that, if there is a $t(n)$-time algorithm that accepts a dense subset of $R_{\mathrm{Kt}}^{n-c\sqrt{n}\log n}$, then the following promise problem can be solved in $\mathsf{DTIME}(O(t(2s(n)) \cdot 2^{\sqrt{s(n)}\log n})) \cap \mathsf{coRTIME}(O(t(2s(n))))$.

YES*: strings x such that $\mathrm{Kt}(x) \leq s$, where $s := s(|x|)$.*

NO*: strings x such that $\mathrm{K}^{t'}(x) > s + c'\sqrt{s}\log n$, where $s := s(|x|)$ and $t' = t(2s) \cdot \mathsf{poly}(|x|)$.*

Lemma 5.4 (see [32])**.** *For any $d, m \leq n, \epsilon > 0$, there exists a black-box pseudorandom generator construction $G^x \colon \{0,1\}^d \to \{0,1\}^m$ that takes a string $x \in \{0,1\}^n$ and satisfies the following.*

1. *For any ϵ-distinguisher $T \colon \{0,1\}^m \to \{0,1\}$ for G^x, it holds that*

$$\mathrm{K}^{t,T}(x) \leq \exp(\ell^2/d) \cdot m + d + O(\log(n/\epsilon)),$$

   where $\ell = O(\log n)$ and $t = \mathsf{poly}(n, 1/\epsilon)$.

2. *$G^x(z)$ is computable in time $\mathsf{poly}(n, 1/\epsilon)$.*

Proof Sketch. The pseudorandom generator construction G^x is defined as $G^x(z) := \mathrm{NW}^{\mathrm{Enc}(x)}(z)$ for any $z \in \{0,1\}^d$, where NW is the Nisan–Wigderson generator [56] that is instantiated with the weak design of [61] and the function whose truth table is $\mathrm{Enc}(x)$ as a candidate hard function. ◻

Proof of Theorem 5.3. Fix $n \in \mathbb{N}$ and an input $x \in \{0,1\}^n$. For simplicity, let $s := s(n)$. Define $d := \sqrt{s}\log n$. Since $s(n) = \omega(\log^2 n)$, we have $4cd \le s$ for a sufficiently large n. Set $m := s + 4cd \le 2s$. Let D be the $t(n)$-time algorithm that accepts a dense subset of $R_{\mathrm{Kt}}^{n-c\sqrt{n}\log n}$.

We claim that there exists a coRP-type algorithm A that uses d random bits and solves the promise problem. The algorithm A operates as follows. Pick a random seed $z \in \{0,1\}^d$, and accept if and only if $D(G^x(z)) = 0$, where the output length of G^x is set to be m. This runs in time $t(m) + \mathsf{poly}(n) \le O(t(2s))$ and can be derandomized in time $O(t(2s)) \cdot 2^d$ by exhaustively trying all the random bits.

It remains to prove the correctness of the algorithm A. Assume that $\mathrm{Kt}(x) \le s$. Then, since $G^x(z)$ is computable in polynomial time, for any $z \in \{0,1\}^d$, we have

$$\mathrm{Kt}(G^x(z)) \le d + s + O(\log n) \le s + 2d \le s + 4cd - 2cd < m - c\sqrt{m}\log n,$$

where, in the last inequality, we used the fact that $\sqrt{m}\log n \le \sqrt{2s}\log n < 2d$. Therefore, $G^x(z) \notin R_{\mathrm{Kt}}^{m-c\sqrt{m}\log m}$, which is rejected by D; hence, A accepts.

Conversely, assume that the probability that A accepts is at least $\frac{3}{4}$. This means that

$$\Pr_{z\sim\{0,1\}^d}[D(G^x(z)) = 0] \ge \frac{3}{4}.$$

Since we have

$$\Pr_{w\sim\{0,1\}^m}[D(w) = 0] \le \frac{1}{2},$$

D is a distinguisher for G^x. By the security of Lemma 5.4, we obtain that, for $t' := \mathsf{poly}(n) \cdot t(m)$,

$$\begin{aligned}\mathrm{K}^{t'}(x) &\le \exp(\ell^2/d) \cdot m + O(\log n)\\ &\le (1 + 2\ell^2/d) \cdot (s + 4cd) + O(\log n)\\ &\le s + O(\sqrt{s}\log n),\end{aligned}$$

which can be bounded above by $s + c'\sqrt{s}\log n$ by choosing a large enough constant c'. Thus, x is not a No instance of the promise problem. Taking the contrapositive, we conclude that any No instance x is rejected by A with probability at least $\frac{1}{4}$. ◻

In the special case when $t(n) = n^{O(1)}$, Theorem 5.3 implies that the Kt vs K^t Problem can be solved in coRP.

Proof Sketch of Theorem 1.15. We claim that the Kt vs K^t Problem can be solved in time $2^{\widetilde{O}(\sqrt{n})}$ under the assumption that MKtP $\in \mathsf{P}$. We use the same proof of Theorem 5.3 for $t(n) := n^{\log n}$ when the parameter $s = \omega(\log^2 n)$; when $s = O(\log^2 n)$, we set $d := \ell^2$ and $m := O(d)$ as in [32]. ◻

Theorem 5.5 (A formal version of Theorem 1.14)**.** *For any constant $0 < \epsilon < 1$, the following are equivalent.*

1. *For some c_1, for any large c_2, there exists no derandomization algorithm for* $\mathsf{DTIME}(2^{c_1\sqrt{n}\log n})$ *that runs in time* $2^{n-c_2\sqrt{n}\log n}$.

2. *For some c_1, for any large c_2, there exists an algorithm running in time $2^{c_1\sqrt{n}\log n}$ that accepts a dense subset of* $R_{\mathrm{Kt}}^{n-c_2\sqrt{n}\log n}$.

3. *For some c_1, for any large c_2,* $\mathrm{MKtP}[n - c_2\sqrt{n}\log n, n-1] \in \mathsf{DTIME}(2^{c_1\sqrt{n}\log n})$.

4. *For some c_1, for any large c_2,* $\mathrm{MKtP}[n^\epsilon, n^\epsilon + c_2\sqrt{n^\epsilon}\log n] \in \mathsf{DTIME}(2^{c_1\sqrt{n^\epsilon}\log n})$.

Proof. (Item 1 $\iff$ Item 2) This equivalence follows from Proposition 5.2.

(Item 2 $\implies$ Item 4) We apply Theorem 5.3 for $t(n) := 2^{c_1\sqrt{n}\log n}$ and $s(n) := n^\epsilon$. Then we obtain a $\mathsf{DTIME}(O(t(2s)) \cdot 2^{\sqrt{s}\log n})$-algorithm A that distinguishes YES instances x such that $\mathrm{Kt}(x) \le s$ from No instances x such that $\mathrm{K}^{t'}(x) > s + c_2'\sqrt{s}\log n$, where $t' = t(2s)\cdot \mathsf{poly}(n)$ and c_2' is some constant. We choose a constant c_1' large enough so that $O(t(2s)) \cdot 2^{\sqrt{s}\log n} \le 2^{c_1'\sqrt{s}\log n}$, which bounds from above the running time of the algorithm A.

We claim that $\mathrm{MKtP}[s, s + c_2''\sqrt{s}\log n]$ is also solved by A for any sufficiently large c_2''. Take any string x such that $\mathrm{Kt}(x) \ge s + c_2''\sqrt{s}\log n$. Since $\mathrm{Kt}(x) \le \mathrm{K}^{t'}(x) + O(\log t') \le \mathrm{K}^{t'}(x) + O(c_1\sqrt{s}\log n)$, it follows that $\mathrm{K}^{t'}(x) \ge s + c_2''\sqrt{s}\log n - O(c_1\sqrt{s}\log n) > s + c_2'\sqrt{s}\log n$, where the last inequality holds for any sufficiently large c_2''; thus, x is rejected by A.

(Item 4 $\implies$ Item 3) By the assumption, there exists a constant c_1 such that, for all large c_2, there exists an algorithm A that solves $\mathrm{MKtP}[n^\epsilon, n^\epsilon + c_2\sqrt{n^\epsilon}\log n]$ in time $2^{c_1\sqrt{n^\epsilon}\log n}$. Define $c_1' := c_1/\epsilon$. For all large c_2', we construct an algorithm B that solves $\mathrm{MKtP}[n - c_2'\sqrt{n}\log n, n-1]$ in time $2^{c_1'\sqrt{n}\log n}$. The algorithm B takes an input x of length n and simulates A on input $x10^{m-n-1}$ for $m := (n - 2c_2\sqrt{n}\log n)^{1/\epsilon}$. The running time of B is at most $2^{c_1\sqrt{m}\log m} \le 2^{c_1/\epsilon\cdot\sqrt{m}\log n} = 2^{c_1'\sqrt{m}\log n}$.

It remains to prove the correctness of B. Take any $x \in \{0,1\}^n$ such that $\mathrm{Kt}(x) \le n - c_2'\sqrt{n}\log n$. Then we have $\mathrm{Kt}(x10^{m-n-1}) \le n - c_2'\sqrt{n}\log n + O(\log n) \le m^\epsilon$ for any large c_2; thus B accepts x. Conversely, take any x such that $\mathrm{Kt}(x) \ge n-1$. Then we have $\mathrm{Kt}(x10^{m-n-1}) \ge n - O(\log n) = m^\epsilon + 2c_2\sqrt{n}\log n - O(\log n) \ge m^\epsilon + c_2\sqrt{m^\epsilon}\log n$; thus, B rejects.

(Item 3 $\implies$ Item 2) This immediately follows from the fact that the complement of $\mathrm{MKtP}[n - c_2\sqrt{n}\log n, n-1]$ is a dense subset of $R_{\mathrm{Kt}}^{n-c_2\sqrt{n}\log n+1}$. $\square$

A Derandomized hardness amplification theorem

We review a simplified proof of derandomized hardness amplification given by Healy, Vadhan and Viola [31], and observe that the parameter shown in Theorem 4.24 can be achieved by slightly modifying the construction.

Theorem 4.24 (Derandomized hardness amplification)**.** *Let $\gamma > 0$ be an arbitrary constant. There exists a hardness amplification procedure* Amp *that takes a function $f\colon \{0,1\}^n \to \{0,1\}$ and parameters $\delta, \epsilon > 0$, and returns a Boolean function $\mathrm{Amp}^f_{\epsilon,\delta}\colon \{0,1\}^{O(n+\log(1/\epsilon))} \to \{0,1\}$ satisfying the following:*

1. *If* $\widetilde{\mathrm{size}}(\mathrm{Amp}^f_{\epsilon,\delta}; 1/2-\epsilon) \leq s$*, then* $\widetilde{\mathrm{size}}(f;\delta) \leq s \cdot \mathsf{poly}(1/\epsilon, 1/\delta)$*.*

2. *For any fixed* $y \in \{0,1\}^{O(n+\log(1/\epsilon))}$*, there exist strings* $v_1, \cdots, v_k \in \{0,1\}^n$ *for some* $k = O(\log(1/\epsilon)/\delta)$ *such that* $\mathrm{Amp}^f_{\epsilon,\delta}(y) = f(v_1) \oplus \cdots \oplus f(v_k)$*. Moreover, if* y *is distributed uniformly at random, for each* $i \in [k]$*,* v_i *is distributed uniformly at random.*

3. $\mathrm{Amp}^f_{\epsilon,\delta}(y)$ *can be computed in* $O(n + \log(1/\delta) + \log(1/\epsilon))$ *space, given* f, y, ϵ *and* δ *as an input.*

First, we explain the construction of our hardness amplification procedure. The construction is a XOR of the nearly disjoint generator ND and a hitter Hit (see [25]). In fact, we need a slightly generalized version of a hitter, for which we show that the same construction of [25] suffices. We note that in the previous works [44, 31], an expander walk was used instead of Hit; this does not give us a nearly optimal construction of a hitter when δ is not a constant.

Lemma A.1. *There exists a* "hitter" Hit *such that, given parameters* $n \in \mathbb{N}, \epsilon, \delta > 0$*, a function* $\mathrm{Hit}_{n,\epsilon,\delta} \colon \{0,1\}^{O(n+\log(1/\epsilon))} \to (\{0,1\}^n)^{k_{n,\epsilon,\delta}}$ *takes a seed of length* $O(n + \log(1/\epsilon))$ *and outputs a list* L *of* $k_{n,\epsilon,\delta} = O(\log(1/\epsilon)/\delta)$ *strings of length* n *such that, for any subsets* $H_1, \cdots, H_{k_{n,\epsilon,\delta}} \subseteq \{0,1\}^n$ *of size* $\geq \delta 2^n$*, with probability at least* $1-\epsilon$*, there exists an index* $i \in [k_{n,\epsilon,\delta}]$ *such that the ith string in the list* L *is in* H_i*. Moreover,* $\mathrm{KS}(L) \leq O(n + \log(1/\epsilon))$*.*

Proof Sketch. We observe that the construction of [25, Appendix C] satisfies the requirement of Lemma A.1. The construction is as follows: First, we take a pairwise independent generator $G_2 \colon \{0,1\}^{O(n)} \to (\{0,1\}^n)^{O(1/\delta)}$. By Chebyshev's inequality, with probability at least $\frac{1}{2}$ over the choice of a seed $r \in \{0,1\}^{O(n)}$, G_2 hits some $H_i, \cdots, H_{i+O(1/\delta)}$, where i is an arbitrary index. Now we take an explicit construction of a constant-degree expander on $2^{O(n)}$ vertices, and generate a random walk $v_1, \cdots, v_\ell$ of length $\ell = O(\log(1/\epsilon))$ over $\{0,1\}^{O(n)}$, which takes random bits of length $O(n + \log(1/\epsilon))$. The output of Hit is defined as the concatenation of $G_2(v_1), \cdots, G_2(v_\ell)$. The correctness follows by using the Expander Random Walk Theorem [25, Theorem A.4]. $\square$

Definition A.2. Let f be a function $f \colon \{0,1\}^n \to \{0,1\}$, and $\epsilon, \delta > 0$ be arbitrary parameters. Let $k := k_{n,\epsilon,\delta}$. Let $\mathrm{ND} \colon \{0,1\}^{O(n)} \to (\{0,1\}^n)^k$ be the nearly disjoint generator (Definition 4.17) defined with the design of Lemma 4.16. We define a generator

$$\mathrm{IW}_{n,\epsilon,\delta} \colon \{0,1\}^{O(n+\log(1/\epsilon))} \to (\{0,1\}^n)^k$$

as

$$\mathrm{IW}_{n,\epsilon,\delta}(x,y) := \mathrm{ND}(x) \oplus \mathrm{Hit}(y).$$

Then we define a hardness amplified version

$$\mathrm{Amp}^f_{\epsilon,\delta} \colon \{0,1\}^{O(n+\log(1/\epsilon))} \to \{0,1\}$$

of f as the function $\mathrm{Amp}^f_{\epsilon,\delta} := f^{\oplus k} \circ \mathrm{IW}_{n,\epsilon,\delta}$.

We proceed to a proof of Theorem 4.24. We recall several notions from [31]. For a subset $H \subseteq \{0,1\}^n$ (which is supposed to be a hard-core set) and a function $f\colon \{0,1\}^n \to \{0,1\}$, we consider a *probabilistic function* $f_H\colon \{0,1\}^n \to \{0,1\}$, i. e., a distribution over Boolean functions, defined as follows: For each $x \in H$, the value $f_H(x)$ is defined as a random bit chosen uniformly at random and independently; For each $x \in \{0,1\}^n \setminus H$, the value $f_H(x)$ is defined as $f(x)$. Assuming that H is indeed a hard-core set for f, the distributions $(x, f(x))$ and $(x, f_H(x))$ where $x \sim \{0,1\}^n$ are computationally indistinguishable. Indeed:

Proposition A.3 (see [70])**.** *Let $H \subseteq \{0,1\}^n$ be an arbitrary subset, $f\colon \{0,1\}^n \to \{0,1\}$ be an arbitrary function, and $\epsilon > 0$ be an arbitrary parameter. If*

$$\Pr_{x\sim\{0,1\}^n}[A(x, f(x)) = 1] - \Pr_{x\sim\{0,1\}^n}[A(x, f_H(x)) = 1] \geq \epsilon,$$

for some function $A\colon \{0,1\}^{n+1} \to \{0,1\}$, then there exists a one-query oracle circuit of size $O(1)$ such that

$$\Pr_{x\sim H}[C^A(x) = f(x)] \geq \frac{1}{2} + \frac{\epsilon}{\delta}.$$

For a probabilistic function $h\colon \{0,1\}^n \to \{0,1\}$, the expected bias of h is defined as $\mathrm{ExpBias}[h] := \mathbb{E}_{x\sim\{0,1\}^n}[\mathrm{Bias}(h(x))]$, where the bias $\mathrm{Bias}(b)$ of a binary random variable $b \in \{0,1\}$ is defined as

$$\mathrm{Bias}(b) = \left|\Pr_b[b=0] - \Pr_b[b=1]\right|.$$

It was shown in [31] that the hardness of $f^{\oplus k} \circ G(z)$ is essentially characterized by the expected bias of $\mathrm{ExpBias}[f_H^{\oplus k} \circ \mathrm{IW}]$, using a property of the nearly disjoint generator ND.

Lemma A.4 ([31, Lemma 5.2 and Lemma 5.12])**.** *Let g be an arbitrary probabilistic function, and $\epsilon > 0$ be arbitrary parameters. Suppose that there exists a function A such that*

$$\Pr_z[A(z) = f^{\oplus k} \circ \mathrm{IW}(z)] \geq \frac{1}{2} + \frac{1}{2}\mathrm{ExpBias}[g^{\oplus k} \circ \mathrm{IW}] + \frac{\epsilon}{2}$$

Then there exists a one-query oracle circuit C of size $O(k \cdot 2^{\gamma n})$ such that

$$\mathbb{E}_x[C^A(x, f(x)) - C^A(x, g(x))] \geq \frac{\epsilon}{2k},$$

where $\gamma > 0$ is an arbitrary constant of Lemma 4.16.

We will apply Lemma A.4 for $g := f_H$. By using a property of the hitter, we show that the expected bias of f_H is small whenever a hard-core set H is large enough.

Lemma A.5. *Let $\epsilon, \delta > 0$ be arbitrary parameters, and $H \subseteq \{0,1\}^n$ be a subset of size at least $\delta 2^n$. Let $k := k_{n,\epsilon,\delta}$. Then* $\mathrm{ExpBias}[f_H^{\oplus k} \circ \mathrm{IW}_{n,\epsilon,\delta}] \leq \epsilon$

Proof. The idea is that when a hitter hits a hard-core set H, the expected bias becomes 0. More specifically, recall that $\mathrm{IW}(x, y)$ is defined as $\mathrm{ND}(x) \oplus \mathrm{Hit}_{n,\epsilon,\delta}(y)$. Fix any x, and define H_i as a shifted version of H: namely, $H_i := \mathrm{ND}(x)_i \oplus H := \{\, \mathrm{ND}(x)_i \oplus h \mid h \in H \,\}$ for every $i \in [k]$. We apply Lemma A.1 for $H_1, \cdots, H_k$. Since $|H_i| = |H| \geq \delta 2^n$ for every $i \in [k]$, by the property of the hitter, there exists some index $i \in [k]$ such that $\mathrm{Hit}_{n,\epsilon,\delta}(y)_i \in H_i = \mathrm{ND}(x)_i \oplus H$ with probability at least $1 - \epsilon$ over the choice of y. In this case, $\mathrm{IW}_{n,\epsilon,\delta}(x, y)_i = \mathrm{ND}(x)_i \oplus \mathrm{Hit}_{n,\epsilon,\delta}(y)_i \in H$, and hence the bias of $f_H(\mathrm{IW}_{n,\epsilon,\delta}(x, y))$ is exactly 0. Therefore,

$$
\begin{aligned}
& \mathrm{ExpBias}[f_H^{\oplus k} \circ \mathrm{IW}_{n,\epsilon,\delta}] \\
\leq\ & \Pr_{x,y}\left[\mathrm{Bias}(f_H^{\oplus k} \circ \mathrm{IW}_{n,\epsilon,\delta}(x, y)) > 0\right] \\
\leq\ & \Pr_{x,y}\left[\, \mathrm{IW}_{n,\epsilon,\delta}(x, y)_i \notin H \text{ for every } i \in [k] \,\right] \ \leq\ \epsilon
\end{aligned}
$$

□

Now we combine Proposition A.3 and Lemmas A.4 and A.5 and obtain the following:

Corollary A.6. *Let $n \in \mathbb{N}$ and $\epsilon, \delta > 0$. Let $H \subseteq \{0, 1\}^n$ be an arbitrary subset of size at least $\delta 2^n$. If some function A satisfies*

$$\Pr_z[A(z) = f^{\oplus k} \circ \mathrm{IW}_{n,\epsilon,\delta}(z)] \geq \frac{1}{2} + \epsilon$$

then there exists a one-query oracle circuit C of size $O(k \cdot 2^{\gamma n})$ such that,

$$\Pr_{x \sim H}[C^A(x) = f(x)] \geq \frac{1}{2} + \frac{\epsilon}{2\delta k}.$$

At this point, we make use of Impagliazzo's hard-core lemma [42]. Equivalently, one can view it as a boosting algorithm (see [47]).

Lemma A.7 (see [42, 47])**.** *Under the condition of Corollary A.6, there exists an oracle circuit C of size $O(k \cdot 2^{\gamma n}) \cdot \mathsf{poly}(k/\epsilon)$ such that*

$$\Pr_{x \sim \{0,1\}^n}[C^A(x) = f(x)] \geq 1 - \delta.$$

Now we take any circuit A of size s such that

$$\Pr_z[A(z) = \mathrm{Amp}^f_{\epsilon,\delta}(z)] \geq \frac{1}{2} + \epsilon.$$

By using Corollary A.6 and Lemma A.7 and replacing each oracle gate by a circuit A, we obtain a circuit C of size $O(k \cdot 2^{\gamma n}) \cdot \mathsf{poly}(k/\epsilon) \cdot s$ such that

$$\Pr_{x \sim \{0,1\}^n}[C(x) = f(x)] \geq 1 - \delta.$$

This completes the proof of Theorem 4.24.

B AC^0 Lower Bounds for MKtP

In this section, we prove that there exists no constant-depth fixed-polynomial-size circuit that computes $\mathrm{MKtP}[O(\log N), N^{o(1)}]$.

Proposition B.1. *For any constants $\alpha < 1, k, d \in \mathbb{N}$, there exists a constant c such that*

$$\mathrm{MKtP}[c \log N, N^{\alpha}] \notin \mathsf{i.o.AC}^0_d(N^k).$$

Previously, [14] used the pseudorandom restriction of Trevisan and Xue [72] to obtain AC^0 lower bounds for $\mathrm{MKtP}[\mathrm{polylog} N]$. Here we make use of a polynomial-time-computable pseudorandom restriction that shrinks AC^0 circuits, which enables us to prove a lower bound for $\mathrm{MKtP}[O(\log N)]$. The following lemma is proved in the context of quantified derandomization.

Lemma B.2 (Goldreich and Wigderson [27]). *For any constants $\alpha < 1$ and $d \in \mathbb{N}$ and any polynomials p, q, there exists a constant k and a polynomial-time algorithm of $O(\log n)$ randomness complexity that produces restrictions on n variables such that the following conditions hold:*

1. *The number of undetermined variables in each restriction is at least $2n^{\alpha}$.*
2. *For any n-input circuit of depth d and size at most $p(n)$, with probability at least $1 - 1/q(n)$, the corresponding restricted circuit depends on at most k variables.*

Proof of Proposition B.1. Fix any polynomial p and a depth d. We claim that, for some constant $c > 0$, there exists no depth-d circuit of size $p(n)$ that computes $\mathrm{MKtP}[c \log n, n^{\alpha}]$ for all large n. Assume, towards a contradiction, that there exists a depth-d circuit C of size $p(n)$ that computes $\mathrm{MKtP}[c \log n, n^{\alpha}]$. We use Lemma B.2 for $q \equiv 2$. Then, there exists a polynomial-time algorithm that produces a pseudorandom restriction ρ that shrinks C to a circuit of size $k = O(1)$ with probability at least $\frac{1}{2}$. Fix one pseudorandom restriction ρ such that the restricted circuit $C \restriction_{\rho}$ is a constant-size circuit.

We claim that C does not compute $\mathrm{MKtP}[c \log n, n^{\alpha}]$. Let σ be the restriction that fixes k variables on which $C \restriction_{\rho}$ depend to 0. Consider $0^n \circ \rho = 0^n \circ \sigma \circ \rho$, i. e., the string obtained by fixing undetermined variables in ρ to 0. Since ρ is generated by a polynomial-time algorithm, there exists a constant c such that $\mathrm{Kt}(0^n \circ \rho) \le c \log n$. Thus, $0^n \circ \sigma \circ \rho$ is a YES instance of $\mathrm{MKtP}[c \log n, n^{\alpha}]$. Since $C \restriction_{\sigma \circ \rho}$ is a constant circuit, we have $1 = C \restriction_{\sigma\circ\rho} (0^n) = C \restriction_{\sigma\circ\rho} (x)$ for any $x \in \{0,1\}^n$. However, for a random $x \in \{0,1\}^n$, by a simple counting argument, it holds that $\mathrm{Kt}(x \circ \sigma \circ \rho) \ge 2n^{\alpha} - k - 1 > n^{\alpha}$ with high probability. Therefore, $x \circ \sigma \circ \rho$ is a No instance of $\mathrm{MKtP}[c \log n, n^{\alpha}]$ for some x, which is a contradiction. ◻

References

[1] LEONARD M. ADLEMAN: Two theorems on random polynomial time. In *Proc. 19th FOCS*, pp. 75–83. IEEE Comp. Soc., 1978. [doi:10.1109/SFCS.1978.37] 19, 37, 47

[2] ERIC ALLENDER, HARRY BUHRMAN, MICHAL KOUCKÝ, DIETER VAN MELKEBEEK, AND DETLEF RONNEBURGER: Power from random strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006. [doi:10.1137/050628994] 14, 15, 17, 30, 32

[3] 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. [doi:10.1137/060664537] 10, 15

[4] ERIC ALLENDER AND SHUICHI HIRAHARA: New insights on the (non-)hardness of circuit minimization and related problems. *ACM Trans. Comput. Theory*, 11(4):27:1–27, 2019. [doi:10.1145/3349616] 15, 30

[5] ERIC ALLENDER AND MICHAL KOUCKÝ: Amplifying lower bounds by means of self-reducibility. *J. ACM*, 57(3):14:1–36, 2010. [doi:10.1145/1706591.1706594] 16

[6] LÁSZLÓ BABAI, LANCE FORTNOW, NOAM NISAN, AND AVI WIGDERSON: *BPP* has subexponential time simulations unless *EXPTIME* has publishable proofs. *Comput. Complexity*, 3(4):307–318, 1993. [doi:10.1007/BF01275486] 3

[7] MANUEL BLUM AND SILVIO MICALI: How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984. [doi:10.1137/0213053] 3

[8] ANDREJ BOGDANOV AND LUCA TREVISAN: On worst-case to average-case reductions for NP problems. *SIAM J. Comput.*, 36(4):1119–1159, 2006. [doi:10.1137/S0097539705446974] 4, 18

[9] ALLAN BORODIN, ALEXANDER A. RAZBOROV, AND ROMAN SMOLENSKY: On lower bounds for read-k-times branching programs. *Comput. Complexity*, 3(1):1–18, 1993. [doi:10.1007/BF01200404] 13, 16

[10] HARRY BUHRMAN, LANCE FORTNOW, AND ADURI PAVAN: Some results on derandomization. *Theory Computing Sys.*, 38(2):211–227, 2005. [doi:10.1007/s00224-004-1194-y] 5, 7, 24, 25

[11] HARRY BUHRMAN, LANCE FORTNOW, AND RAHUL SANTHANAM: Unconditional lower bounds against advice. In *Proc. 36th Internat. Colloq. on Automata, Languages, and Programming (ICALP'09)*, pp. 195–209. Springer, 2009. [doi:10.1007/978-3-642-02927-1_18] 7

[12] HARRY BUHRMAN AND STEVEN HOMER: Superpolynomial circuits, almost sparse oracles and the exponential hierarchy. In *Proc. 12th Found. Softw. Techn. Theoret. Comp. Sci. Conf. (FSTTCS'92)*, pp. 116–127. Springer, 1992. [doi:10.1007/3-540-56287-7_99] 28

[13] MARCO L. CARMOSINO, RUSSELL IMPAGLIAZZO, VALENTINE KABANETS, AND ANTONINA KOLOKOLOVA: Learning algorithms from Natural Proofs. In *Proc. 31st Comput. Complexity Conf. (CCC'16)*, pp. 10:1–24. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016. [doi:10.4230/LIPIcs.CCC.2016.10] 15

[14] LIJIE CHEN, SHUICHI HIRAHARA, IGOR CARBONI OLIVEIRA, JÁN PICH, NINAD RAJGOPAL, AND RAHUL SANTHANAM: Beyond Natural Proofs: Hardness magnification and locality. *J. ACM*, 69(4):25:1–49, 2022. Preliminary version in ITCS'20. [doi:10.1145/3538391] 11, 16, 54

[15] LIJIE CHEN, CE JIN, AND R. RYAN WILLIAMS: Hardness magnification for all sparse NP languages. In *Proc. 60th FOCS*, pp. 1240–1255. IEEE Comp. Soc., 2019. [doi:10.1109/FOCS.2019.00077] 16

[16] KUAN CHENG AND WILLIAM M. HOZA: Hitting sets give two-sided derandomization of small space. *Theory of Computing*, 18(21):1–32, 2022. Preliminary version in CCC'20. [doi:10.4086/toc.2022.v018a021] 13

[17] MAHDI CHERAGHCHI, SHUICHI HIRAHARA, DIMITRIOS MYRISIOTIS, AND YUICHI YOSHIDA: One-tape Turing machine and branching program lower bounds for MCSP. *Theory Computing Sys.*, (10113-9):32 pages, 2022. Preliminary version in MFCS'21. [doi:10.1007/s00224-022-10113-9] 13

[18] MAHDI CHERAGHCHI, VALENTINE KABANETS, ZHENJIAN LU, AND DIMITRIOS MYRISIOTIS: Circuit lower bounds for MCSP from local pseudorandom generators. *ACM Trans. Comput. Theory*, 12(3):21:1–27, 2020. Preliminary version in ICALP'19. [doi:10.1145/3404860] 11

[19] SHIMON EVEN, ALAN L. SELMAN, AND YACOV YACOBI: The complexity of promise problems with applications to public-key cryptography. *Inform. Control*, 61(2):159–173, 1984. [doi:10.1016/S0019-9958(84)80056-X] 2

[20] JOAN FEIGENBAUM AND LANCE FORTNOW: Random-self-reducibility of complete sets. *SIAM J. Comput.*, 22(5):994–1005, 1993. [doi:10.1137/0222061] 4

[21] MAGNUS GAUSDAL FIND, ALEXANDER GOLOVNEV, EDWARD A. HIRSCH, AND ALEXANDER S. KULIKOV: A better-than-$3n$ lower bound for the circuit complexity of an explicit function. In *Proc. 57th FOCS*, pp. 89–98. IEEE Comp. Soc., 2016. [doi:10.1109/FOCS.2016.19] 9

[22] MICHAEL A. FORBES AND ZANDER KELLEY: Pseudorandom generators for read-once branching programs, in any order. In *Proc. 59th FOCS*, pp. 946–955. IEEE Comp. Soc., 2018. [doi:10.1109/FOCS.2018.00093] 13

[23] LANCE FORTNOW: Comparing notions of full derandomization. In *Proc. 16th IEEE Conf. on Comput. Complexity (CCC'01)*, pp. 28–34. IEEE Comp. Soc., 2001. [doi:10.1109/CCC.2001.933869] 11

[24] ODED GOLDREICH: On promise problems: A survey. In ODED GOLDREICH, ARNOLD L. ROSENBERG, AND ALAN L. SELMAN, editors, *Theoretical Computer Science, Essays in Memory of Shimon Even*, pp. 254–290. Springer, 2006. [doi:10.1007/11685654_12] 2

[25] ODED GOLDREICH: A sample of samplers: A computational perspective on sampling. In ODED GOLDREICH, editor, *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pp. 302–332. Springer, 2011. [doi:10.1007/978-3-642-22670-0_24] 51

[26] ODED GOLDREICH, SHAFI GOLDWASSER, AND SILVIO MICALI: How to construct random functions. *J. ACM*, 33(4):792–807, 1986. [doi:10.1145/6490.6503] 30

[27] ODED GOLDREICH AND AVI WIGDERSON: On derandomizing algorithms that err extremely rarely. In *Proc. 46th STOC*, pp. 109–118. ACM Press, 2014. [doi:10.1145/2591796.2591808] 54

[28] ODED GOLDREICH AND AVI WIGDERSON: On the size of depth-three boolean circuits for computing multilinear functions. In ODED GOLDREICH, editor, *Comput. Complexity and Property Testing*, LNCS 12050, pp. 41–86. Springer Cham, 2020. (2013). [ECCC:TR13-043] 10

[29] JOHAN HÅSTAD: *Computational limitations of small-depth circuits*. Ph. D. thesis, MIT, 1986. Available on author's website, published as a book by MIT Press, 1987. 10

[30] JOHAN HÅSTAD, RUSSELL IMPAGLIAZZO, LEONID A. LEVIN, AND MICHAEL LUBY: A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. [doi:10.1137/S0097539793244708] 8, 22, 28, 29

[31] ALEXANDER HEALY, SALIL P. VADHAN, AND EMANUELE VIOLA: Using nondeterminism to amplify hardness. *SIAM J. Comput.*, 35(4):903–931, 2006. [doi:10.1137/S0097539705447281] 41, 50, 51, 52

[32] SHUICHI HIRAHARA: Non-black-box worst-case to average-case reductions within NP. In *Proc. 59th FOCS*, pp. 247–258. IEEE Comp. Soc., 2018. [doi:10.1109/FOCS.2018.00032] 4, 5, 15, 18, 25, 30, 31, 48, 49

[33] SHUICHI HIRAHARA: Characterizing average-case complexity of PH by worst-case meta-complexity. In *Proc. 61st FOCS*, pp. 50–60. IEEE Comp. Soc., 2020. [doi:10.1109/FOCS46700.2020.00014] 7

[34] SHUICHI HIRAHARA: Non-disjoint promise problems from meta-computational view of pseudorandom generator constructions. In *Proc. 35th Comput. Complexity Conf. (CCC'20)*, pp. 20:1–47. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020. [doi:10.4230/LIPIcs.CCC.2020.20] 1

[35] SHUICHI HIRAHARA: Unexpected hardness results for Kolmogorov complexity under uniform reductions. In *Proc. 52nd STOC*, pp. 1038–1051. ACM Press, 2020. [doi:10.1145/3357713.3384251] 5, 7, 8, 15, 22

[36] SHUICHI HIRAHARA, IGOR CARBONI OLIVEIRA, AND RAHUL SANTHANAM: NP-hardness of Minimum Circuit Size Problem for OR-AND-MOD circuits. In *Proc. 33rd Comput. Complexity Conf. (CCC'18)*, pp. 5:1–31. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. [doi:10.4230/LIPIcs.CCC.2018.5] 15

[37] SHUICHI HIRAHARA AND RAHUL SANTHANAM: On the average-case complexity of MCSP and its variants. In *Proc. 32nd Comput. Complexity Conf. (CCC'17)*, pp. 7:1–20. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. [doi:10.4230/LIPIcs.CCC.2017.7] 11, 15, 30

[38] SHUICHI HIRAHARA AND OSAMU WATANABE: Limits of Minimum Circuit Size Problem as oracle. In *Proc. 31st Comput. Complexity Conf. (CCC'16)*, pp. 18:1–20. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016. [doi:10.4230/LIPIcs.CCC.2016.18] 7, 15

[39] SHUICHI HIRAHARA AND OSAMU WATANABE: On nonadaptive security reductions of hitting set generators. In *Proc. 24th Internat. Conf. on Randomization and Computation (RANDOM'20)*, pp. 15:1–14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020. [doi:10.4230/LIPIcs.APPROX/RANDOM.2020.15] 4

[40] JOHN M. HITCHCOCK AND ADURI PAVAN: On the NP-completeness of the Minimum Circuit Size Problem. In *Proc. 35th Found. Softw. Techn. Theoret. Comp. Sci. Conf. (FSTTCS'15)*, pp. 236–245. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015. [doi:10.4230/LIPIcs.FSTTCS.2015.236] 7

[41] RUSSELL IMPAGLIAZZO: Hard-core distributions for somewhat hard problems. In *Proc. 36th FOCS*, pp. 538–545. IEEE Comp. Soc., 1995. [doi:10.1109/SFCS.1995.492584] 3

[42] RUSSELL IMPAGLIAZZO: A personal view of average-case complexity. In *Proc. 10th IEEE Conf. Structure in Complexity Theory (SCT'95)*, pp. 134–147. IEEE Comp. Soc., 1995. [doi:10.1109/SCT.1995.514853] 53

[43] RUSSELL IMPAGLIAZZO: Relativized separations of worst-case and average-case complexities for NP. In *Proc. 26th IEEE Conf. on Comput. Complexity (CCC'11)*, pp. 104–114. IEEE Comp. Soc., 2011. [doi:10.1109/CCC.2011.34] 4

[44] RUSSELL IMPAGLIAZZO AND AVI WIGDERSON: $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proc. 29th STOC*, pp. 220–229. ACM Press, 1997. [doi:10.1145/258533.258590] 3, 11, 12, 17, 19, 41, 51

[45] VALENTINE KABANETS AND JIN-YI CAI: Circuit minimization problem. In *Proc. 32nd STOC*, pp. 73–79. ACM Press, 2000. [doi:10.1145/335305.335314] 15

[46] RICHARD M. KARP AND RICHARD J. LIPTON: Turing machines that take advice. *Enseign. Math.*, 28:191–209, 1982. [doi:10.5169/seals-52237] 9, 33

[47] ADAM R. KLIVANS AND ROCCO A. SERVEDIO: Boosting and hard-core set construction. *Machine Learning*, 51(3):217–238, 2003. [doi:10.1023/A:1022949332276] 53

[48] 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. [doi:10.1137/S0097539700389652] 3, 13, 35, 37

[49] KER-I KO: On the complexity of learning minimum time-bounded Turing machines. *SIAM J. Comput.*, 20(5):962–986, 1991. [doi:10.1137/0220059] 7

[50] LEONID A. LEVIN: Randomness conservation inequalities; Information and independence in mathematical theories. *Inform. Control*, 61(1):15–37, 1984. [doi:10.1016/S0019-9958(84)80060-1] 9, 32

[51] LEONID A. LEVIN: Average case complete problems. *SIAM J. Comput.*, 15(1):285–286, 1986. [doi:10.1137/0215020] 4

[52] WILLIAM J. MASEK: Some NP-complete set covering problems. Master's thesis, MIT CS Lab, 1978. Cited in the NP-completeness column by David Johnson. 15

[53] DYLAN M. MCKAY, CODY D. MURRAY, AND R. RYAN WILLIAMS: Weak lower bounds on resource-bounded compression imply strong separations of complexity classes. In *Proc. 51st STOC*, pp. 1215–1225. ACM Press, 2019. [doi:10.1145/3313276.3316396] 16

[54] CODY D. MURRAY AND R. RYAN WILLIAMS: On the (non) NP-hardness of computing circuit complexity. *Theory of Computing*, 13(4):1–22, 2017. [doi:10.4086/toc.2017.v013a004] 7

[55] NOAM NISAN: Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992. [doi:10.1007/BF01305237] 12

[56] NOAM NISAN AND AVI WIGDERSON: Hardness vs randomness. *J. Comput. System Sci.*, 49(2):149–167, 1994. [doi:10.1016/S0022-0000(05)80043-1] 3, 23, 32, 37, 38, 48

[57] IGOR CARBONI OLIVEIRA, JÁN PICH, AND RAHUL SANTHANAM: Hardness magnification near state-of-the-art lower bounds. *Theory of Computing*, 17(11):1–38, 2021. Preliminary version in CCC'19. [doi:10.4086/toc.2021.v017a011] 16

[58] IGOR CARBONI OLIVEIRA AND RAHUL SANTHANAM: Hardness magnification for natural problems. In *Proc. 59th FOCS*, pp. 65–76. IEEE Comp. Soc., 2018. [doi:10.1109/FOCS.2018.00016] 10, 15, 16

[59] RAFAIL OSTROVSKY: One-way functions, hard on average problems, and statistical zero-knowledge proofs. In *Proc. 6th IEEE Conf. Structure in Complexity Theory (SCT'91)*, pp. 133–138. IEEE Comp. Soc., 1991. [doi:10.1109/SCT.1991.160253] 31

[60] ADURI PAVAN, RAHUL SANTHANAM, AND N. V. VINODCHANDRAN: Some results on average-case hardness within the Polynomial Hierarchy. In *Proc. 26th Found. Softw. Techn. Theoret. Comp. Sci. Conf. (FSTTCS'06)*, pp. 188–199. Springer, 2006. [doi:10.1007/11944836_19] 6

[61] RAN RAZ, OMER REINGOLD, AND SALIL P. VADHAN: Extracting all the randomness and reducing the error in Trevisan's extractors. *J. Comput. System Sci.*, 65(1):97–128, 2002. [doi:10.1006/jcss.2002.1824] 48

[62] ALEXANDER A. RAZBOROV: Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Math. Notes. Acad. Sci. USSR (English translation)*, 41(4):333–338, 1987. [doi:10.1007/BF01137685] 11

[63] ALEXANDER A. RAZBOROV AND STEVEN RUDICH: Natural Proofs. *J. Comput. System Sci.*, 55(1):24–35, 1997. [doi:10.1006/jcss.1997.1494] 15, 16, 17, 30

[64] OMER REINGOLD, LUCA TREVISAN, AND SALIL P. VADHAN: Notions of reducibility between cryptographic primitives. In *Proc. Theory of Cryptography Conf. (TCC'04)*, pp. 1–20. Springer, 2004. [doi:10.1007/978-3-540-24638-1_1] 18

[65] RAHUL SANTHANAM: Pseudorandomness and the Minimum Circuit Size Problem. In *Proc. 11th Innovations in Theoret. Comp. Sci. Conf. (ITCS'20)*, pp. 68:1–26. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020. [doi:10.4230/LIPIcs.ITCS.2020.68] 15

[66] RONEN SHALTIEL AND EMANUELE VIOLA: Hardness amplification proofs require majority. *SIAM J. Comput.*, 39(7):3122–3154, 2010. [doi:10.1137/080735096] 11

[67] MADHU SUDAN: Decoding of Reed Ssolomon codes beyond the error-correction bound. *J. Complexity*, 13(1):180–193, 1997. [doi:10.1006/jcom.1997.0439] 22

[68] MADHU SUDAN, LUCA TREVISAN, AND SALIL P. VADHAN: Pseudorandom generators without the XOR lemma. *J. Comput. System Sci.*, 62(2):236–266, 2001. [doi:10.1006/jcss.2000.1730] 3, 4, 12, 15, 35, 42, 43

[69] 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. [doi:10.1109/MAHC.1984.10036] 15

[70] LUCA TREVISAN: List-decoding using the XOR lemma. In *Proc. 44th FOCS*, pp. 126–135. IEEE Comp. Soc., 2003. [doi:10.1109/SFCS.2003.1238187] 52

[71] LUCA TREVISAN AND SALIL P. VADHAN: Pseudorandomness and average-case complexity via uniform reductions. *Comput. Complexity*, 16(4):331–364, 2007. [doi:10.1007/s00037-007-0233-x] 4

[72] LUCA TREVISAN AND TONGKE XUE: A derandomized Switching Lemma and an improved derandomization of AC0. In *Proc. 28th IEEE Conf. on Comput. Complexity (CCC'13)*, pp. 242–247. IEEE Comp. Soc., 2013. [doi:10.1109/CCC.2013.32] 54

[73] SALIL P. VADHAN: An unconditional study of Computational Zero Knowledge. *SIAM J. Comput.*, 36(4):1160–1214, 2006. [doi:10.1137/S0097539705447207] 31

[74] SALIL P. VADHAN: Pseudorandomness. *Found. Trends Theor. Comp. Sci.*, 7(1–3):1–336, 2012. [doi:10.1561/0400000010] 22, 23, 24, 42

[75] LESLIE G. VALIANT AND VIJAY V. VAZIRANI: NP is as easy as detecting unique solutions. *Theoret. Comput. Sci.*, 47:85–93, 1986. [doi:10.1016/0304-3975(86)90135-0] 2

[76] EMANUELE VIOLA: The complexity of constructing pseudorandom generators from hard functions. *Comput. Complexity*, 13(3–4):147–188, 2005. [doi:10.1007/s00037-004-0187-1] 4, 37

[77] ANDREW CHI-CHIH YAO: Theory and applications of trapdoor functions (extended abstract). In *Proc. 23rd FOCS*, pp. 80–91. IEEE Comp. Soc., 1982. [doi:10.1109/SFCS.1982.45] 3, 29

[78] SERGEY YEKHANIN: Locally decodable codes. *Found. Trends Theor. Comp. Sci.*, 6(3):139–255, 2012. [doi:10.1561/0400000030] 42

AUTHOR

Shuichi Hirahara
Associate professor
Principles of Informatics Research Division
National Institute of Informatics
Tokyo, Japan
s_hirahara@nii.ac.jp
https://researchmap.jp/shuichi.hirahara?lang=en

ABOUT THE AUTHOR

SHUICHI HIRAHARA is an Associate Professor at the National Institute of Informatics, Japan. He obtained his Ph. D. from the University of Tokyo under the supervision of Hiroshi Imai in 2019. After joining the National Institute of Informatics as an Assistant Professor, he was promoted to an Associate Professor in 2022. He is interested in average-case complexity, meta-complexity, and Kolmogorov complexity.