Electronic Colloquium on Computational Complexity, Report No. 119 (2022)

# NP-Hardness of Learning Programs and Partial MCSP

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

August 24, 2022

## Abstract

A long-standing open question in computational learning theory is to prove NP-hardness of learning efficient programs, the setting of which is in between proper learning and improper learning. Ko (COLT'90, SICOMP'91) explicitly raised this open question and demonstrated its difficulty by proving that there exists no relativizing proof of NP-hardness of learning programs. In this paper, we overcome Ko's relativization barrier and prove NP-hardness of learning programs under randomized polynomial-time many-one reductions. Our result is provably non-relativizing, and comes somewhat close to the parameter range of improper learning: We observe that mildly improving our inapproximability factor is sufficient to exclude Heuristica, i.e., show the equivalence between average-case and worst-case complexities of NP.

We also make progress on another long-standing open question of showing NP-hardness of the Minimum Circuit Size Problem (MCSP). We prove NP-hardness of the partial function variant of MCSP as well as other meta-computational problems, such as the problems MKTP$^*$ and MINKT$^*$ of computing the time-bounded Kolmogorov complexity of a given partial string, under randomized polynomial-time reductions.

Our proofs are algorithmic information (a.k.a. Kolmogorov complexity) theoretic. We utilize black-box pseudorandom generator constructions, such as the Nisan–Wigderson generator, as a one-time encryption scheme secure against a program which "does not know" a random function. Our key technical contribution is to quantify the "knowledge" of a program by using conditional Kolmogorov complexity and show that no small program can know many random functions.

ISSN 1433-8092

## Contents

**1 Introduction** **1**
1.1 PAC Learning . . . . . 1
1.2 Meta-Complexity . . . . . 3

**2 An Overview of Our Proofs** **7**
2.1 NP-Hardness of Learning Programs . . . . . 7
2.2 NP-Hardness of Partial Variants of Meta-computatioal Problems . . . . . 10

**3 Open Problems** **12**

**4 Preliminaries** **13**
4.1 Kolmogorov Complexity . . . . . 13
4.2 Secret Sharing Scheme . . . . . 14

**5 Collective Minimum Monotone Satisfying Assignment Problem** **15**

**6 Algorithmic Information Extraction Lemmas** **18**

**7 NP-Hardness of Learning Programs** **22**

**8 NP-Hardness of Partial MCSP** **24**

**A Weak Learning by Small Hypotheses in Heuristica** **33**

**B MINLT and Ko's Relativization Barrier** **36**

**C NP-Hardness of Learning Small Circuits** **38**

## 1 Introduction

The two main results of this paper are NP-hardness of MINLT [Ko91] and the partial function variant of MCSP [KC00], both of which are of central importance in computational learning theory and meta-complexity theory. In the following two subsections, we present the background of the two theories as well as our results, respectively.

### 1.1 PAC Learning

Ever since Valiant [Val84] introduced the notion of PAC learning, classifying the complexity of PAC learning has been a fundamental and central question in computational learning theory. The task of PAC learning is parameterized by a *concept class* $\mathcal{C}$ and a *hypothesis class* $\mathcal{H}$. Informally, a class $\mathcal{C}$ is said to be *PAC learnable by* $\mathcal{H}$ if there exists an efficient algorithm $L$ such that for every distribution $\mathcal{D}$ and for every concept $c \in \mathcal{C}$, given sufficiently many random samples $(x_1, c(x_1)), \dots, (x_m, c(x_m))$, where each $x_i$ is drawn from $\mathcal{D}$ independently, the learning algorithm $L$ outputs a hypothesis $h \in \mathcal{H}$ such that $\Pr_{x \sim \mathcal{D}}[h(x) = c(x)] \geq 1 - \delta$ for a given parameter $\delta$. A fundamental theorem of computational learning theory [BEHW87; BP92; Sch90] shows that, for a sufficiently large hypothesis class $\mathcal{H}$, PAC learning is equivalent to Occam learning, the latter of which can be formulated as a search problem in NP.[1] An outstanding open question is to prove NP-hardness of PAC learning of linear-sized circuits by polynomial-sized circuits, which would classify the complexity of PAC learning as "NP-complete".

NP-hardness of PAC learning has been proved in the case of *proper learning*, i.e., the case when $\mathcal{C} = \mathcal{H}$. Pitt and Valiant [PV88] showed NP-hardness of learning $k$-term DNF by $k$-term DNF. This was extended to NP-hardness of learning linear-sized DNF formulas by the polynomial-sized disjunction of halfspaces [ABFKP08]; i.e., $\mathcal{C} = \{\text{DNF formulas}\}$ and $\mathcal{H} = \text{OR} \circ \{\text{halfspaces}\}$. Note that, as the hypothesis class $\mathcal{H}$ becomes larger, it becomes increasingly harder to prove NP-hardness. Consider, for example, the class $\mathsf{NC}^1$ of fan-in-2 circuits of logarithmic depth. Intuitively, PAC learning of $\mathsf{NC}^1$ by $\mathsf{NC}^1$ appears to be much harder than PAC learning of DNFs by DNFs, since $\mathsf{NC}^1$ is larger than the class of DNF formulas. Despite this intuition, NP-hardness of learning linear-sized $\mathsf{NC}^1$ by polynomial-sized $\mathsf{NC}^1$ is unknown.

The opposite of proper learning is called *improper learning*, in which there is no restriction on the hypothesis class $\mathcal{H}$, except that $\mathcal{H}$ must be evaluated in polynomial time. As already noted in the seminal work of Valiant [Val84], it is possible to prove hardness of improper PAC learning under cryptographic assumptions. A recent exciting line of research (e.g., [DS16; Vad17; DV21]) starting from Daniely, Linial, and Shalev-Shwartz [DLS14] shows that specific average-case hardness assumptions of NP already imply hardness of improper PAC learning. However, there is a fundamental obstacle that prevents us from proving NP-hardness of improper PAC learning. Applebaum, Barak, and Xiao [ABX08] showed that NP-hardness of improper PAC learning cannot be proved by nonadaptive reductions unless the polynomial hierarchy collapses. They also proved that NP-hardness of improper PAC learning excludes Pessiland [Imp95] from Impagliazzo's five worlds, i.e., shows the equivalence between the existence of one-way functions and average-case hardness of NP. Moreover, a partial converse to [DLS14] was recently proved by Hirahara and Nanashima [HN21]: PAC learning with respect to distributions samplable by polynomial-size circuits is feasible under the assumption that NP is easy on average. Thus, the complexity of improper PAC

[1]The task of Occam learning is to output a short description of a hypothesis that is consistent with given samples $(x_1, c(x_1)), \dots, (x_m, c(x_m))$ for an unknown concept $c$.

learning is intimately related to average-case complexity of $\mathsf{NP}$. In particular, proving $\mathsf{NP}$-hardness of PAC learning (with respect to $\mathsf{P/poly}$ samplable distributions) also excludes Heuristica from Impagliazzo's five worlds [Imp95], i.e., shows the equivalence between worst-case and average-case complexities of $\mathsf{NP}$. These previous works indicate the importance and, at the same time, the difficulty of proving $\mathsf{NP}$-hardness of improper PAC learning.

Ko [Ko91] raised the question of classifying the complexity of learning efficient programs, i.e., PAC learning by $\mathcal{H}$, where the hypothesis class $\mathcal{H}$ is the class of efficient programs. Arguably, this is the most general hypothesis class: By the fundamental principle of Kolmogorov complexity [Kol65], the Kolmogorov complexity of a string $x$, i.e., the length of a shortest program that prints $x$, is a lower bound on the length of any decodable encoding of the string $x$ up to an $O(1)$ additive term. Similarly, Ko observed that representing a function by a program is the most succinct way of representing a function by any algorithm up to an $O(1)$ additive term. In particular, programs can represent a function more succinctly than circuits.

Ko formulated the task of learning efficient programs by introducing a problem called MINLT, which is a decision version of Occam learning for efficient programs. The input of MINLT consists of $((x_1, b_1), \dots, (x_m, b_m); 1^t, 1^s)$, where $x_i \in \{0,1\}^n$ and $b_i \in \{0,1\}$ for some $n \in \mathbb{N}$, and the objective is to decide whether there exists a $t$-time program $M$ of size $s$ that is *consistent* with the given samples $(x_1, b_1), \dots, (x_m, b_m)$, i.e., $M(x_i) = b_i$ for every $i$. Since the complexity of MINLT "appears very difficult to classify precisely" [Ko91], Ko gave a formal evidence for this statement, by proving that there exists no relativizing proof that shows $\mathsf{NP}$-hardness of MINLT. A *relativizing proof*, the notion of which was introduced by Baker, Gill, and Solovay [BGS75] to argue the difficulty of resolving the $\mathsf{P}$ versus $\mathsf{NP}$ question, refers to a proof of a complexity-theoretic statement that can be generalized to the statements in the presence of arbitrary oracles. Although there are several non-relativizing proof techniques (see, e.g., [BFT98; AW09]), vast majority of complexity-theoretic proofs are relativizing; thus, proving a non-relativizing statement is highly challenging and of major importance in complexity theory. Indeed, we are not aware of any previous non-relativizing result in complexity theory for which relativization barriers were presented three decades ago.[2]

In this work, we overcome Ko's relativization barrier and resolve the long-standing open problem of proving $\mathsf{NP}$-hardness of MINLT.

**Theorem 1.1.** *Under randomized polynomial-time one-query reductions, it is* $\mathsf{NP}$*-hard to distinguish the following cases, given as input a size parameter* $s \in \mathbb{N}$ *and a distribution* $\mathcal{E}$ *such that* $\mathrm{supp}(\mathcal{E}) \subseteq \{0,1\}^n \times \{0,1\}$ *for some* $n \in \mathbb{N}$.[3]

**Yes:** *There exists a polynomial-time program* $M$ *of size* $s$ *such that*

$$\Pr_{(x,b)\sim\mathcal{E}}[M(x) = b] = 1.$$

*Moreover,* $M$ *computes a linear function over* $\mathrm{GF}(2)$.

[2]As discussed in [AW09, Section 9], there are several non-relativizing statements that exploit the subtleties in defining oracle access mechanism. Those include statements on small depth circuits and bounded-space Turing machines. It is controversial whether such results are "truly" non-relativizing. By contrast, Theorem 1.1 can be naturally relativized by using the standard definition of oracle Turing machines.

[3]A distribution $\mathcal{E}$ is represented by a circuit $C$ such that the output $C(r)$ of $C$ over a uniformly random string $r \sim \{0,1\}^{|C|}$ is identical to $\mathcal{E}$. The support of $\mathcal{E}$ is denoted by $\mathrm{supp}(\mathcal{E})$.

**No:** *For any program $M$ of size $s \cdot n^{\epsilon}$,*

$$\Pr_{(x,b)\sim\mathcal{E}}[M(x) = b] \le \frac{1}{2} + 2^{-n^{1-\delta}}$$

*Here, $\delta > 0$ is an arbitrary constant and $\epsilon = 1/(\log\log n)^{O(1)}$.*

A couple of remarks are in order. First, the problem considered in Theorem 1.1 can be reduced to MINLT by drawing $m$ samples $(x_1, b_1), \ldots, (x_m, b_m)$ from the distribution $\mathcal{E}$ for $m = O(s)$; thus, it also shows NP-hardness of MINLT under randomized reductions as an immediate corollary. Our result is *provably* non-relativizing. Although Ko [Ko91] stated his relativization barrier for deterministic reductions, it can be extended to randomized reductions. See Appendix B for the details. Second, Theorem 1.1 refers to the decision version of PAC learning of a concept class $\mathcal{C}$ by a hypothesis class $\mathcal{H}$, where $\mathcal{C}$ is the class of efficient programs of size $s$ (that compute a linear function) and $\mathcal{H}$ is the class of *time-unbounded* programs of size $s \cdot n^{\epsilon}$. Note that the decision version is reducible to the standard search version, as long as $\mathcal{H}$ can be evaluated in polynomial time; thus, another corollary of Theorem 1.1 is NP-hardness of PAC learning of $\mathcal{C}$ by $\mathcal{H}'$, where $\mathcal{H}' \subseteq \mathcal{H}$ is the class of polynomial-time programs of size $s \cdot n^{\epsilon}$. Let us emphasize that, in the No case, even *time-unbounded* programs fail to output $b$ on input $x$ for a random sample $(x, b)$ drawn from $\mathcal{E}$. Note that finding one hypothesis that is consistent with $\mathcal{E}$ is easy by using Gaussian elimination; Theorem 1.1 shows that deciding whether such a hypothesis can be compressed as a small time-unbounded program is NP-hard. Third, the probability $\frac{1}{2} + 2^{-n^{1-\delta}}$ in the No case is close to the optimal, as a trivial hypothesis that always outputs either 0 or 1 agrees with the samples from $\mathcal{E}$ with probability at least $\frac{1}{2}$. Although there have been many works on NP-hardness of learning with large error (e.g., [ABFKP08; KS08; GKS10; KS11; FGKP09; FGRW12]), most results show NP-hardness of learning with error $\frac{1}{2} - \epsilon$ for a constant $\epsilon > 0$; we are not aware of any previous result that achieves the exponentially small correlation bound of $2^{-n^{1-\delta}}$.

Theorem 1.1 comes somewhat close to the range of parameters for which proof techniques on improper PAC learning can be applied. For example, using the proof techniques of [HN21], we observe that improving the inapproximability factor $n^{\epsilon}$ to $1.01n$ is sufficient to exclude Heuristica; see Appendix A for the details.

## 1.2 Meta-Complexity

A problem closely related to PAC learning is the *Minimum Circuit Size Problem* [KC00] (MCSP). The problem MCSP is, given a Boolean function $f : \{0,1\}^n \to \{0,1\}$ represented as the truth table of length $2^n$ as well as a size parameter $s \in \mathbb{N}$, to decide whether there exists a circuit of size $s$ that computes $f$. The study of MCSP is said to date back to as early as 1960s [Tra84]. Although it is easy to see that MCSP $\in$ NP, it is a long-standing open problem to prove NP-hardness of MCSP; indeed, it is reported in [AKRR11] that Levin [Lev73] delayed the publication of his seminal work on the theory of NP-completeness because he hoped to prove NP-completeness of MCSP.

More generally, MCSP is an example of *meta-computational problems.* Meta-complexity refers to the computational complexity of problems that themselves ask for complexity. The aforementioned work of Ko [Ko91] introduced the problem MINKT of deciding whether a given string $x$ can be printed by a $t$-time program of size $s$, given $(x, 1^t, 1^s)$ as input. Similarly, MKTP is the problem of deciding whether each bit of a given string $x$ can be computed by a $t$-time program of size $s$

for some $(t, s)$ such that $t + s \leq \theta$, given $(x, \theta)$ as input [ABKMR06]. All of these problems are meta-computational: MCSP asks the circuit complexity of a given string; MINKT asks the time complexity of printing a given string (i.e., time-bounded Kolmogorov complexity); MKTP asks the trade-off between the time complexity and the size complexity of printing a given string. Technically, MKTP is often considered as a convenient surrogate of MCSP for which many theorems that are not known to hold for MCSP can be proved (e.g., [HS17; AH19]). Meta-complexity has recently received significant attention because of its connection to diverse areas of theoretical computer science, including learning theory [CIKK16; HN21], average-case complexity [Hir18; Hir21], cryptography [IL90; San20; LP20], and circuit lower bounds [OS18; CHOPRS20]; see the survey of Allender [All21] for a broad overview.

Although none of the meta-computational problems mentioned above is shown to be $\mathsf{NP}$-hard, there has been recent substantial progress on $\mathsf{NP}$-hardness of meta-computational problems. For restricted circuit classes $\mathfrak{C} \in \{\mathrm{DNF}, \mathrm{DNF} \circ \mathrm{XOR}, \mathsf{AC}^0 \text{ formulas}\}$, the corresponding versions of MCSP (denoted by $\mathfrak{C}$-MCSP) were shown to be $\mathsf{NP}$-hard [Mas79; AHMPS08; HOS18; Ila20b]. More recently, Ilango [Ila21] proved that the formula variant of MCSP is hard under the Exponential-Time Hypothesis (ETH) [IPZ01].

A well-trodden path in the proofs of $\mathsf{NP}$-hardness of variants of MCSP consists of two steps: The first step is to prove $\mathsf{NP}$-hardness of the partial function variants of MCSP, which are often denoted by MCSP$^*$. The input of MCSP$^*$ consists of a partial function $f \colon \{0,1\}^n \to \{0,1,*\}$ (encoded as a string of length $2^{O(n)}$) and a size parameter $s \in \mathbb{N}$, and the task is to decide whether there exists a circuit of size $s$ that computes $f(x)$ on input $x$ such that $f(x) \neq *$. The second step is to present a reduction from the the partial variants of MCSP to MCSP. For example, the proof of $\mathsf{NP}$-hardness of DNF-MCSP presented in [AHMPS08] is given by composing two reductions, a reduction from $\mathsf{NP}$ to DNF-MCSP$^*$ and a reduction from DNF-MCSP$^*$ to DNF-MCSP. Other works [HOS18; Ila21] follow the same paradigm. Thus, proving $\mathsf{NP}$-hardness of partial variants of meta-computational problems serves as a milestone toward $\mathsf{NP}$-hardness of the total versions of meta-computational problems. In fact, Levin's seminal paper [Lev73] presented six $\mathsf{NP}$-complete problems; the second problem shown to be $\mathsf{NP}$-complete was DNF-MCSP$^*$. Since the introduction of the theory of $\mathsf{NP}$-completeness [Coo71; Lev73], it has been a long-standing open problem to extend Levin's $\mathsf{NP}$-completeness result to MCSP$^*$.

Why is it difficult to prove $\mathsf{NP}$-hardness of meta-computational problems? The difficulty of proving $\mathsf{NP}$-hardness of MCSP is closely related to the inability of proving explicit circuit lower bounds. Kabanets and Cai [KC00] showed that if MCSP is $\mathsf{NP}$-hard under deterministic "natural" reductions, then $\mathsf{EXP} \not\subseteq \mathsf{P/poly}$.[4] Their proof techniques are applicable to MCSP$^*$ as well. Since $\mathsf{EXP} \not\subseteq \mathsf{P/poly}$ is a major open question in complexity theory, this suggests that proving $\mathsf{NP}$-hardness of MCSP$^*$ would be quite difficult, at least under deterministic reductions. A subsequent line of work improved this barrier result of [KC00] to more general types of reductions, such as "non-natural" reductions [MW17], nonadaptive deterministic reductions [HP15], adaptive deterministic reductions [HW16; SS20]. The intuition behind all of these results is that if there is a many-one reduction from SAT to MCSP$^*$, then the reduction, given an unsatisfiable formula as input, must produce the truth table of a function $f$ with high circuit complexity, which implies a circuit lower bound for $\mathsf{EXP}$.

To avoid this barrier, one may be tempted to consider randomized reductions. A randomized

[4]A *natural reduction to* MCSP is a reduction such that the size parameter in the output of the reduction depends only on the input length.

reduction can easily produce a function with high circuit complexity, as a random function has high circuit complexity with high probability; thus, the barriers mentioned above do not apply to randomized reductions. Unfortunately, Hirahara and Watanabe [HW16] present evidence that suggests the difficulty of proving NP-hardness of MCSP even under randomized reductions: They observed that most reductions to MCSP are *oracle-independent*, i.e., they can be generalized to the Minimum $A$-Oracle Circuit Size Problem (denoted by $\mathrm{MCSP}^A$) for every oracle $A$. Then, they showed that there is no one-query randomized polynomial-time reduction from NP to $\mathrm{MCSP}^A$ for some oracle $A$ unless the polynomial hierarchy collapses.

Yet another approach for bypassing the barriers is to use exponential-time reductions. Ilango [Ila20b] recently bypassed the barrier of [KC00] and proved ETH-hardness of $\mathrm{MCSP}^*$ under deterministic reductions. The intuition behind this result is that lower bounds against $O(n)$-size circuits can be proved using gate elimination techniques; since it is already known that there exists an explicit function that cannot be computed by a circuit of size $cn$ for a constant $c$, the barrier of [KC00] does not apply. Such proof techniques of using deterministic reductions, however, are unlikely to be extended to NP-hardness of $\mathrm{MCSP}^*$, as the barriers of [KC00] come into play. We also note that it would be extremely difficult to prove NP-hardness of $\mathrm{MKTP}^*$ and $\mathrm{MINKT}^*$ under deterministic reductions, as there are few proof techniques of showing a lower bound of the time-bounded Kolmogorov complexity of an explicit string. Here, $\mathrm{MKTP}^*$ and $\mathrm{MINKT}^*$ are the partial variants of MKTP and MINKT, which ask the minimum of the time-bounded Kolmogorov complexity of $y$ over all the strings $y \in \{0,1\}^*$ that are *consistent with*[5] a given partial string $x \in \{0,1,*\}^*$ (see Definition 8.4 for the formal definitions).

Using our proof techniques of showing NP-hardness of PAC learning, we circumvent the barrier of [KC00] by means of randomized reductions and prove NP-hardness of partial variants of several meta-computational problems.

**Theorem 1.2.** *Under randomized polynomial-time one-query reductions, it is* NP*-hard to distinguish the following two cases, given a partial function* $f\colon \{0,1\}^n \to \{0,1,*\}$ *(encoded as a string over* $\{0,1,*\}$ *of length* $2^n$*), a size parameter* $s \in \mathbb{N}$*, and a distribution* $\mathcal{D}$ *over* $f^{-1}(\{0,1\})$*:*

**Yes:** *There exist an* $s$*-time*[6] *program* $M$ *of size* $s$ *and a circuit* $C$ *of size* $\frac{s}{\log s}$ *and depth* $O(\log s)$ *such that*

$$\Pr_{x\sim\mathcal{D}}[M(x) = f(x)] = 1 \quad \textit{and} \quad \Pr_{x\sim\mathcal{D}}[C(x) = f(x)] = 1.$$

**No:** *For every program* $M$ *of size* $s \cdot n^{\epsilon}$ *and for every circuit* $C$ *of size* $\frac{s}{\log s} \cdot n^{\epsilon}$*,*

$$\Pr_{x\sim\mathcal{D}}[M(x) = f(x)] \le \frac{1}{2} + n^{-\epsilon} \quad \textit{and} \quad \Pr_{x\sim\mathcal{D}}[C(x) = f(x)] \le \frac{1}{2} + n^{-\epsilon}.$$

*Here,* $\epsilon > 0$ *is a universal constant. In particular,* $\mathrm{MCSP}^*$*,* $\mathrm{NC}^1$-$\mathrm{MCSP}^*$*,* $\mathrm{MKTP}^*$*, and* $\mathrm{MINKT}^*$ *are* NP*-hard under randomized polynomial-time reductions. Moreover, these problems are* NP*-hard to approximate within a factor of* $(\log N)^{\epsilon}$ *on inputs of length* $N$*.*

[5]We say that a string $y \in \{0,1\}^n$ is *consistent with* a partial string $x \in \{0,1,*\}^n$ if every bit of $x$ is equal to either $*$ or the bit of $y$ at the corresponding position.

[6]Here, the time complexity of $M$ is measured as in KT-complexity; that is, we assume that a program is given oracle access to the description of $M$; see Definition 4.1.

Our proofs are inspired by a line of research [AHMPS08; HOS18; Ila20a; ILO20; ACMTV21; LP22; Hir22], which developed "top-down" approaches toward showing $\mathsf{NP}$-hardness of MCSP.[7] Theorem 1.2 is proved by fundamentally different proof techniques from the previous result of [Ila20b] and significantly improves it in the following perspectives: (1) The time complexity of a reduction is improved from exponential to polynomial. (2) We prove hardness of approximation, whereas [Ila20a] does not. (3) We prove $\mathsf{NP}$-hardness of $\mathrm{MKTP}^*$ and $\mathrm{MINKT}^*$. (4) In addition, by slightly modifying the $\mathsf{NP}$-hardness reduction of $\mathrm{MCSP}^*$,[8] we also prove $\mathsf{NP}$-hardness of the average-case variant of MCSP called AveMCSP [San20], which asks the *average-case* circuit complexity of a given *total* function $f\colon \{0,1\}^n \to \{0,1\}$ with respect to the uniform distribution.

The size parameter $s$ in Theorem 1.2 is exponential in the input length $n$ (i.e, $s = 2^{\Theta(n)}$), which is inevitable unless $\mathsf{NP}$ can be solved in randomized sub-exponential time.[9] One would be surprised at the strong circuit lower bound of the No case: It shows that no circuit of exponential size can compute the function $f$. Given our poor knowledge on circuit lower bounds for explicit functions,[10] how can one prove such a strong circuit lower bound? Of course, the function $f$ is not explicit, but naively, one would expect that any proof of $\mathsf{NP}$-hardness of $\mathrm{MCSP}^*$ would require a complicated analysis on exponential-size circuits. This is indeed the main difficulty that prevented researchers from proving $\mathsf{NP}$-hardness of $\mathrm{MCSP}^*$ (and formalized as the barrier of [KC00] in the case of deterministic reductions). Surprisingly, we prove the circuit lower bound by using *almost nothing* about circuits: The circuit lower bound simply follows from the fact that no program of size $s$ can approximate $f$. Since a circuit of size $s'$ can be encoded as a binary string of length $O(s' \log s')$, we obtain a circuit lower bound of $s' \geq \Omega(s/\log s)$. By contrast, the construction of the circuit $C$ in the Yes case is more complicated.

All the previous $\mathsf{NP}$-hardness results for $\mathfrak{C}$-MCSP are proved by different reductions for each circuit class $\mathfrak{C}$. Ideally, one would like to prove $\mathsf{NP}$-hardness of $\mathfrak{C}$-MCSP for any sufficiently large circuit class $\mathfrak{C}$ by a single reduction. Unfortunately, this is not possible: The aforementioned work of [HW16] shows that MCSP is not reducible to $\mathrm{MCSP}^A$ for some oracle $A$ (unless MCSP is easy), despite that $A$-oracle circuits are more powerful than circuits. Surprisingly, Theorem 1.2 indicates that the same is not true for partial variants of MCSP. Our reduction is in fact oracle-independent in the sense that our reduction also shows $\mathsf{NP}$-hardness of $\mathrm{MKTP}^{*A}$ for every oracle $A$.[11] This indicates that the negative result of [HW16] for MCSP is unlikely to be extended to the partial variant $\mathrm{MCSP}^*$, despite that the negative result of [KC00] can be easily extended to $\mathrm{MCSP}^*$.

The proof of Theorem 1.2 is obtained by optimizing the reduction of Theorem 1.1. Note that the $\mathsf{NP}$-hardness reduction of Theorem 1.2 takes an $\mathsf{NP}$ instance of length $N$ and produces the truth table of a partial function $f\colon \{0,1\}^n \to \{0,1,*\}$, which is of length $2^{O(n)}$, in time $N^{O(1)}$; thus, we must have $n = O(\log N)$. A high-level idea of the proof of Theorem 1.2 is that the input length $n$ of the distribution $\mathcal{E}$ in Theorem 1.1 can be optimized to be as small as $O(\log N)$.

[7] In more detail, based on his $\mathsf{NP}$-hardness result of a conditional variant of MCSP, Ilango [Ila20a] proposed an approach of showing hardness of MCSP "from above", as opposed to "bottom-up" approaches of [AHMPS08; HOS18; Ila20b; Ila21], which try to show $\mathsf{NP}$-hardness of $\mathfrak{C}$-MCSP for larger and larger classes $\mathfrak{C}$. Theorem 1.2 realizes such a top-down approach successfully.

[8] Specifically, we replace $f(x) = *$ with a uniformly random bit $f(x) \sim \{0,1\}$ in the $\mathsf{NP}$-hardness proof of $\mathrm{MCSP}^*$. See Theorem 8.7 for the details.

[9] This follows from the fact that $\mathrm{MCSP}^*$ with size parameter $s(n)$ can be solved in time $2^{O(s(n)\log s(n))}$ by a brute-force search.

[10] For example, it is a central open problem to prove $\mathsf{E}^{\mathsf{NP}} \not\subseteq \mathsf{SIZE}(O(n))$. A function $f \in \mathsf{E}^{\mathsf{NP}}$ is said to be *explicit*.

[11] For a technical reason, our reduction may not prove $\mathsf{NP}$-hardness of $\mathrm{MCSP}^{*A}$.

## 2 An Overview of Our Proofs

In this section, we outline our proofs. At a very high level, our technical contribution is to develop an *algorithmic information (Kolmogorov complexity) theoretic* proof technique for showing lower bounds of the size of programs. We assume familiarity with the notion of Kolmogorov complexity and secret sharing scheme; we encourage the readers unfamiliar with these notions to read Section 4 first.

**Notation** $[n]$ denotes $\{1, \dots, n\}$. For $s_1, \dots, s_n$ and a subset $T \subseteq [n]$, let $s_T$ denote $(s_{i_1}, \dots, s_{i_m})$, where $T = \{i_1 < \cdots < i_m\}$.

### 2.1 NP-Hardness of Learning Programs

We present an overview of the proof of Theorem 1.1, which shows $\mathsf{NP}$-hardness of learning programs. We reduce the Minimum Monotone Satisfying Assignment (MMSA) problem [ABMP01] to the problem of learning programs. The input of MMSA consists of a monotone formula $\varphi$ on $n$ variables and a parameter $\theta \in \mathbb{N}$. The task is to decide whether there exists a satisfying assignment $\alpha \colon [n] \to \{0,1\}$ of $\varphi$ such that $\sum_{i=1}^{n} \alpha(i) \le \theta$. It is known to be $\mathsf{NP}$-hard to approximate MMSA to within a factor of $n^{1/(\log\log n)^{O(1)}}$ [DS04; DHK15].

To reduce MMSA to the learning problem, we use a secret sharing scheme $(\mathrm{Share}(\varphi, \text{-}), \mathrm{Rec}(\varphi, \text{-}))$ for a monotone formula $\varphi$. We say that a subset $T \subseteq [n]$ is *authorized* if the characteristic function $\chi_T \colon [n] \to \{0,1\}$ of $T$ satisfies the formula $\varphi$. The secret sharing scheme allows us to share a secret $b \in \{0,1\}$ among $n$ parties so that any authorized set of parties can reconstruct the secret, whereas no unauthorized set of parties can obtain any information of the secret. At a high level, our reduction produces a distribution $\mathcal{E}$, from which $(x, b)$ is sampled as follows: A random secret $b \sim \{0,1\}$ is shared as $\mathrm{Share}(\varphi, b) =: (s_1, \dots, s_n)$, where each $s_i$ is the share given to the $i$-th party, and then each share $s_i$ is "hidden" in an input $x$. Our main technical contribution is to develop a way of hiding the shares in an input $x$ so that large programs can read the hidden shares from $x$, whereas small programs cannot read many hidden shares.

A key tool for hiding shares in an input is a (black-box) pseudorandom generator construction [TV07]. There are many pseudorandom generator constructions in the literature, such as the Nisan–Wigderson pseudorandom generator [NW94]. Although most pseudorandom generator constructions can be used for our purpose,[12] for the sake of simplicity, we use a simple pseudorandom generator construction called a *$k$-wise direct product generator* $\mathrm{DP}_k \colon \{0,1\}^\lambda \times \{0,1\}^{\lambda \times k} \to \{0,1\}^{\lambda k + k}$ [Hir20b], which is defined as follows. Given a string $f \in \{0,1\}^\lambda$ (regarded as a row vector over $\mathrm{GF}(2)$) and a $\lambda \times k$ matrix $z$ over $\mathrm{GF}(2)$, the output $\mathrm{DP}_k(f; z)$ is defined to be $(z, f \cdot z)$, where $f \cdot z$ denotes the multiplication of a vector $f$ and a matrix $z$ over $\mathrm{GF}(2)$. The pseudorandom generator construction $\mathrm{DP}_k$ satisfies the following "reconstruction" property [Hir20a]: If there exists a function $D$ that $\epsilon$-distinguishes the output distribution $\mathrm{DP}_k(f; \text{-})$ from the uniform distribution, i.e.,

$$\left| \Pr_{z \sim \{0,1\}^{\lambda k}} [D(\mathrm{DP}_k(f; z)) = 1] - \Pr_{w \sim \{0,1\}^{\lambda k + k}} [D(w) = 1] \right| \ge \epsilon,$$

[12] One property of a pseudorandom generator that is required for our reduction is *seed-extending* [KMS12], i.e., a pseudorandom generator is secure even if the seed is included in the output. This property is used in the proof of the completeness of our reduction.

then we have

$$\mathrm{K}^D(f) \le k + O(\log(\lambda k/\epsilon)). \tag{1}$$

In other words, $\mathrm{DP}_k(f;\text{-})\colon \{0,1\}^{\lambda k} \to \{0,1\}^{\lambda k+k}$ is a pseudorandom generator secure against an algorithm $D$ if $f$ is a "hard" function in the sense that $f$ cannot be described using approximately $k$ bits of information and oracle access to $D$. We can think of the pseudorandom generator $\mathrm{DP}_k(f;\text{-})$ as a one-time encryption scheme secure against any algorithm $D$ that "does not know" $f$ in the sense that $\mathrm{K}^D(f) \gg k$. Besides its simplicity, a useful property of $\mathrm{DP}_k(f;\text{-})$ is that $\mathrm{DP}_k(f;\text{-})$ is a linear function for any fixed $f$, which enables us to show the linearity property of the Yes case in Theorem 1.1.

We now present the details of the reduction from MMSA to the learning problem. Let $(\varphi, \theta)$ be an instance of MMSA. We choose $n$ strings $f_1, \ldots, f_n \sim \{0,1\}^\lambda$ uniformly at random. Each string $f_i$ is associated with the $i$-th variable of $\varphi$. Using the strings $f_1, \ldots, f_n$, we define an example distribution $\mathcal{E} = \mathcal{E}(f_1, \ldots, f_n)$ as follows. To sample an example $(x, b) \in \{0,1\}^m \times \{0,1\}$ distributed according to $\mathcal{E}$, we choose a secret $b \sim \{0,1\}$ randomly. We share the secret $b$ among $n$ parties using the secret sharing scheme for $\varphi$; let $(s_1, \ldots, s_n) := \mathrm{Share}(\varphi, b)$ be the set of shares and let $k \in \mathbb{N}$ be the length of each share. The idea is to hide these shares in the input $x$ so that any algorithm that "knows" $f_i$ can read the $i$-th share, whereas any algorithm that "does not know" $f_i$ cannot read the $i$-th share. To this end, we let $\xi_i := (z_i, (f_i \cdot z_i) \oplus s_i) = \mathrm{DP}_k(f_i; z_i) \oplus (0^{\lambda k}, s_i)$ for every $i \in [n]$, where $z_i \sim \{0,1\}^{\lambda k}$. Here, "$\oplus$" denotes the bitwise XOR. Then, we define $x := (\xi_1, \ldots, \xi_n)$. This completes the description of how to sample $(x, b)$ from the example distribution $\mathcal{E}$. In fact, this is a complete description of the reduction of Theorem 1.1. The difficulty lies in the proof of the correctness of the reduction, which we sketch below.

It is not hard to see the completeness of the reduction. Assume that there exists a satisfying assignment $\alpha$ of $\varphi$ such that $\sum_{i=1}^n \alpha(i) \le \theta$. Let $T$ denote the set of indices $i \in [n]$ such that $\alpha(i) = 1$. The fact that the set $T$ is authorized motivates us to define the following program $M$: The program $M$ takes $\{f_i \mid i \in T\}$ as hard-wired input. Given an input $x = (\xi_1, \ldots, \xi_n)$, the program $M$ lets $(z_i, \eta_i) := \xi_i$, defines $s_i := (f_i \cdot z_i) \oplus \eta_i$, reconstructs the secret $b$ using the reconstruction procedure $\mathrm{Rec}(\varphi, \text{-})$ for the shares $\{s_i \mid i \in T\}$, and outputs $b \in \{0,1\}$. Since $T$ is authorized, it is guaranteed that the secret $b \in \{0,1\}$ can be reconstructed in this way. The size of $M$ is approximately at most $\sum_{i \in T} |f_i| = |T| \cdot \lambda \le \theta\lambda$.

To see the soundness of the reduction, we first clarify the condition under which the randomized reduction is successful. The condition is that $\mathrm{K}(f_T) \gtrsim |T| \cdot \lambda$ for every $T$, which happens with high probability over the random choice of $f_1, \ldots, f_n$ by a simple counting argument. (Throughout this section, an approximate inequality $a \lesssim b$ can be understood as $a \le (1 + o(1)) \cdot b$, where $o(1)$ approaches to 0 as the parameter $\lambda$ increases.) Now, assuming that there exists no authorized set $T$ of size $\theta$, we claim that no program $M$ of size $\theta\lambda/2$ can output $b$ on input $x$ with high probability over $(x, b) \sim \mathcal{E}$. Our key technical lemma, which we call an *algorithmic information extraction lemma*, is informally stated as follows.

**Lemma 2.1** (informal; see Lemma 6.1)**.** *Let $f_1, \ldots, f_n \in \{0,1\}^\lambda$ be strings such that $\mathrm{K}(f_T) \gtrsim |T| \cdot \lambda$ for every $T \subseteq [n]$, where $\lambda$ is sufficiently large. Let $M$ be a program of size $|M|$. Then, there exists a set $B \subseteq [n]$ such that $|B| \lesssim |M|/\lambda$ and*

$$\Pr\bigl[M(X_B, U_{[n]\setminus B}) = 1\bigr] \approx \Pr\bigl[M(X_B, X_{[n]\setminus B}) = 1\bigr],$$

*where $X_i$ is the random variable identical to $\mathrm{DP}_k(f_i; z_i)$ for a random choice of $z_i \sim \{0,1\}^{nk}$ and $U_i$ is identical to the uniform distribution over $\{0,1\}^{\lambda k+k}$.*

This lemma shows that one can "extract" a small set $B$ from a program $M$ such that $\mathrm{DP}_k(f_i; \text{-})$ looks pseudorandom against $M$ for every $i \in [n] \setminus B$. In particular, for every $i \in [n] \setminus B$, the program $M$ cannot read $s_i$ from $\xi_i = \mathrm{DP}_k(f_i; z_i) \oplus s'_i$, where $s'_i$ denotes $(0^{\lambda k}, s_i)$. Moreover, we have $|B| \lesssim |M|/\lambda \le \theta/2$, which implies that $B$ is not authorized. By the privacy of the secret sharing scheme, the shares $s_B$ and the secret $b \sim \{0,1\}$ are statistically independent; thus, we obtain

$$\Pr\Big[M(X_B \oplus s'_B, U_{[n]\setminus B} \oplus s'_{[n]\setminus B}) = b\Big] = \Pr\big[M(X_B \oplus s'_B, U_{[n]\setminus B}) = b\big] = \frac{1}{2}.$$

It follows from the algorithmic information extraction lemma that[13]

$$\begin{aligned}\Pr_{(x,b)\sim\mathcal{E}}[M(x) = b] &= \Pr\Big[M(X_B \oplus s'_B, X_{[n]\setminus B} \oplus s'_{[n]\setminus B}) = b\Big] \\ &\approx \Pr\Big[M(X_B \oplus s'_B, U_{[n]\setminus B} \oplus s'_{[n]\setminus B}) = b\Big] = \frac{1}{2},\end{aligned}$$

as desired.

It remains to prove the algorithmic information extraction lemma. We formalize the notion that $M$ "knows" $f_i$ by using conditional Kolmogorov complexity: We say that $M$ *knows* $f_i$ if

$$\mathrm{K}(f_i \mid M) \le \theta$$

for a threshold $\theta := 2nk$. The intuition behind this definition is that if $M$ contains a lot of information about $f_i$, it should be possible to describe $f_i$ using a few bits of information.[14] Now, we define $B$ to be the set of indices $i \in [n]$ such that $M$ knows $f_i$. It can be shown that $|B|$ is small:

$$|B| \cdot \lambda - |M| \lesssim \mathrm{K}(f_B) - |M| \lesssim \mathrm{K}(f_B \mid M) \lesssim \sum_{i\in B} \mathrm{K}(f_i \mid M) \le |B| \cdot \theta, \tag{2}$$

where the first inequality is due to the assumption, the second inequality holds by the definition of conditional Kolmogorov complexity, the third inequality holds because $f_B = (f_i \mid i \in B)$ can be described by programs describing $f_i$ for all $i \in B$, and the last inequality holds by the definition of $B$. If we choose a sufficiently large $\lambda$ so that $\theta = o(\lambda)$ (e.g., $\lambda := \theta^2$), we obtain

$$|B| \cdot \lambda \cdot (1 - o(1)) \lesssim |M|,$$

which implies $|B| \lesssim |M|/\lambda$, as desired. We now prove that $M$ cannot distinguish $X_i = \mathrm{DP}_k(f_i; z_i)$ from $U_i$ for every $i \in [n] \setminus B$. This can be proved by a standard (but careful) hybrid argument. To illustrate the idea, let us assume that $B = \{2\}$ and $n = 2$. In this case, we show

$$\Pr_{z_1,z_2}[M(\mathrm{DP}_k(f_1; z_1), \mathrm{DP}_k(f_2; z_2)) = 1] \approx \Pr_{U_1,z_2}[M(U_1, \mathrm{DP}_k(f_2; z_2)) = 1].$$

Assume, toward a contradiction, that this approximate equality fails. Our goal is to prove $\mathrm{K}(f_1 \mid M) \le \theta$, which contradicts that $M$ does not know $f_1$ (i.e., $1 \notin B$). We use the reconstruction

[13] The argument is not precise, especially because we need to argue that the set $B$ does not depend on the secret $b$ and the shares $s_{[n]}$. To address this issue, the algorithmic information extraction lemma will be stated for a program that takes an advice string $\alpha$, which includes the information about the secret and the shares; see Lemma 6.1 for details.

[14] In terms of the mutual information $\mathrm{I}(f_i : M) := \mathrm{K}(f_i) - \mathrm{K}(f_i \mid M)$ between $f_i$ and $M$, the condition that $M$ knows $f_i$ means that $\mathrm{I}(f_i : M) \approx \mathrm{K}(f_i)$.

property of $\mathrm{DP}_k$. However, it is problematic to apply the property of Eq. (1) naively: Applying Eq. (1), we would get $\mathrm{K}^D(f_1) \lesssim k$, where $D$ is a function that outputs $M(w, \mathrm{DP}_k(f_2; z_2))$ on input $w$ and random bits $z_2$. Since $D$ can be simulated using $M$ and $f_2$, we get

$$\mathrm{K}(f_1 \mid M, f_2) \lesssim \mathrm{K}^D(f_1) \lesssim k.$$

We need to get rid of $f_2$ from the condition of $\mathrm{K}(f_1 \mid M, f_2)$. To this end, we observe that $D$ is in fact a randomized algorithm that takes an $a$-bit advice string that depends on randomness for some small $a \in \mathbb{N}$ (independent of $\lambda$); the notion of such advice is introduced by Trevisan and Vadhan [TV07]. Specifically, $D$ takes $w$ and random bits $z_2$ and outputs $M(w, \mathrm{DP}_k(f_2; z_2)) = M(w, (z_2, f_2 \cdot z_2))$; the key insight is that this function can be computed with the $k$-bit advice string $f_2 \cdot z_2 \in \{0,1\}^k$; we do not need the full description of $f_2 \in \{0,1\}^\lambda$ to compute $D$. Using a pseudorandom generator (computable by an exhaustive search), we observe that the reconstruction property of Eq. (1) actually shows

$$\mathrm{K}(f_1 \mid M) \lesssim k + a$$

for any randomized algorithm $M$ that takes $a$ bits of Trevisan–Vadhan advice. In our case, we have $a \le (n-1) \cdot k$; thus, we can show $\mathrm{K}(f_1 \mid M) \lesssim nk \le \theta/2$. This implies that $M$ knows $f_1$, which contradicts $1 \notin B$.

## 2.2 NP-Hardness of Partial Variants of Meta-computatioal Problems

We now update the proof above to a proof of Theorem 1.2, i.e., $\mathsf{NP}$-hardness of the partial function variants of meta-computational problems, such as $\mathrm{NC}^1$-$\mathrm{MCSP}^*$, $\mathrm{MCSP}^*$, $\mathrm{MKTP}^*$, and $\mathrm{MINKT}^*$. Specifically, our goal is to reduce an instance of an $\mathsf{NP}$-complete problem to an example distribution $\mathcal{E}$ such that $\mathrm{supp}(\mathcal{E}) \subseteq \{0,1\}^{O(\log n)} \times \{0,1\}$, where $n$ is the length of an instance of the origianl $\mathsf{NP}$-complete problem. Enumerating all the elements in the support $\mathrm{supp}(\mathcal{E})$ of the distribution $\mathcal{E}$, we obtain a partial function $f\colon \{0,1\}^{O(\log n)} \to \{0,1,*\}$, which yields a reduction to partial variants of meta-computational problems. Note that the reduction of Theorem 1.1 produces a partial function $f\colon \{0,1\}^{n^{O(1)}} \to \{0,1,*\}$. We need to exponentially reduce the input length of $f$.

We start by inspecting the $\mathsf{NP}$-hardness reduction to the MMSA problem. Dinur and Safra [DS04] showed a generic approximation-preserving reduction from any MaxCSP (Constraint Satisfaction Problem) to MMSA. An instance of MaxCSP consists of the set $\Psi = \{C_1, \ldots, C_m\}$ of constraints over $n$ variables taking a value in an alphabet $\Sigma$. Each constraint $C_j$ depends on $D$ variables, where $D = O(1)$. The reduction of [DS04] reduces such an instance to a depth-3 monotone formula $\varphi$ such that $\varphi = \bigwedge_{j \in [m]} \varphi_j$, where each depth-2 subformula $\varphi_j$ "checks" that the constraint $C_j$ is satisfied. Since each constraint $C_j$ depends on a constant number $D$ of variables, each subformula $\varphi_j$ also depend on a small number of variables. The main idea of reducing the input length in our reduction is to exploit this "locality" of $\varphi_j$.

We introduce the Collective Minimum (Weight) Monotone Satisfying Assignment (CMMSA) problem, which generalizes the MMSA problem. An instance of CMMSA consists of a collection $\mathit{\Phi} = \{\varphi_1, \ldots, \varphi_m\}$ of monotone DNF formulas over $n$ variables, a weight function $w\colon [n] \to \mathbb{N}$, and a threshold $\theta$. The task is to distinguish (1) the Yes case in which there exists an assignment $\alpha\colon [n] \to \{0,1\}$ such that the weight $\sum_{i=1}^n \alpha(i) \cdot w(i)$ is at most $\theta$ and $\Pr_{\varphi \sim \mathit{\Phi}}[\varphi(\alpha) = 1] = 1$ from (2) the No case in which for every assignment of weight $\Delta^\epsilon \cdot \theta$, it holds that $\Pr_{\varphi \sim \mathit{\Phi}}[\varphi(\alpha) = 1] < \Delta^{-\epsilon}$, where $\epsilon > 0$ is a constant and $\Delta$ is a parameter such that the number of literals in the DNF formula $\varphi_j$ is at most $\Delta$ for every $j \in [m]$. Using a PCP system [DFKRS11] developed in the research line on the Sliding Scale Conjecture [BGLR94], we observe that CMMSA is $\mathsf{NP}$-hard for some constant $\epsilon > 0$ and for a parameter $\Delta := (\log n)^{1/2}$, by applying the reduction of [DS04]. We note that the weight function $w$ in CMMSA is used for a technical reason; thus, for the purpose of simplicity, we assume that $w \equiv 1$ in this proof overview.

Now, our goal is to reduce CMMSA to MINKT$^*$. To this end, we exploit the "locality" of CMMSA: Each formula $\varphi_j$ in $\Phi$ depends on at most $\Delta$ variables out of $n$ variables. The reduction is simliar to the reduction of Theorem 1.1. We choose $f_1, \ldots, f_n \sim \{0,1\}^\lambda$ randomly. We define a distribution $\mathcal{E} = \mathcal{E}(f_1, \ldots, f_n)$ as follows: Choose a secret $b \sim \{0,1\}$ and $\varphi_j \sim \Phi$ randomly. Using the secret sharing scheme for $\varphi_j$, we share $b$ among at most $\Delta$ parties $v_1, \ldots, v_\Delta \in [n]$. Let $(s_1, \ldots, s_\Delta) = \mathrm{Share}(\varphi_j, b)$ be the shares distributed to the parties $v_1, \ldots, v_\Delta$, respectively. We define an input $x$ to be $(j, \mathrm{DP}_k(f_{v_1}; z_1) \oplus s'_1, \ldots, \mathrm{DP}_k(f_{v_\Delta}; z_\Delta) \oplus s'_\Delta)$, where $s'_i := (0^{\lambda k}, s_i)$. The output of the distribution $\mathcal{E}$ is defined to be $(x, b)$. Just as in the proof of Theorem 1.1, it is possible to prove the following: If there is an assignment $\alpha\colon [n] \to \{0,1\}$ that satisfies all the formulas $\varphi_j$ in $\Phi$, then there exists a program of size $\lesssim \sum_{i=1}^n \alpha(i) \cdot \lambda$ (that takes $\{f_i \mid \alpha(i) = 1\}$ as hard-wired input) that computes $b$ on input $x$ for every $(x, b) \in \mathrm{supp}(\mathcal{E})$. If there is no assignment of small weight that satisfies a $\Delta^{-\epsilon}$-fraction of $\Phi$, then there is no small program that computes $b$ on input $x$ with probability $1/2 + 2\Delta^{-\epsilon}$ over a random choice of $(x, b) \sim \mathcal{E}$.

The only issue is that the length of the seeds $z_1, \ldots, z_\Delta$ is large, which prevents us from getting $|x| = O(\log n)$. The length of each $z_i$ is $\lambda k$, which is always greater than the length of $f_i$. This comes from the fact that $\mathrm{DP}_k$ is instantiated with the Hadamard encoding, which maps a string $f_i$ of length $\lambda$ to a string of length $2^\lambda$. Instead, if we instantiate a $k$-wise direct product generator with an error correcting code that maps a string of length $\lambda$ to a string of length $\lambda^{O(1)}$, we can reduce the length of seed $z_i$ to $k \cdot O(\log \lambda)$ (as in [Hir20b]). This optimization is not still sufficient to get $|x| = O(\log n)$, as the seeds $z_1, \ldots, z_\Delta$ are independent. To further reduce the seed length, we use the Nisan–Wigderson pseudorandom generator construction [NW94]. Nisan and Wigderson [NW94] developed a way of generating correlated seeds $z_{S_1}, \ldots, z_{S_\Delta}$ from a short seed $z$; they used it to construct a pseudorandom generator with short seed length based on an average-case hard function. Using the Nisan–Wigderson pseudorandom generator construction NW, we construct the input $x$ as $(j, z, \mathrm{NW}(\mathrm{Enc}(f_{v_1}); z_{S_1}) \oplus s_1, \ldots, \mathrm{NW}(\mathrm{Enc}(f_{v_\Delta}); z_{S_\Delta}) \oplus s_\Delta)$, where $\mathrm{Enc}\colon \{0,1\}^\lambda \to \{0,1\}^{\lambda^{O(1)}}$ is an error-correcting code. It can be shown that the input length $|x|$ is $O(\log n)$. This completes an overview of the proof of $\mathsf{NP}$-hardness of MINKT$^*$.

Showing $\mathsf{NP}$-hardness of MCSP$^*$ requires additional ideas. We need to argue that there is a small circuit $C$ that takes $x = (j, z, \mathrm{NW}(\mathrm{Enc}(f_{v_1}); z_{S_1}) \oplus s_1, \ldots, \mathrm{NW}(\mathrm{Enc}(f_{v_\Delta}); z_{S_\Delta}) \oplus s_\Delta)$ as input, reads shares from the input, and reconstructs a secret. This can be done by a $\mathsf{poly}(\lambda)$-time algorithm by computing $\mathrm{Enc}(f_{v_i})$ from a hard-wired input $f_{v_i}$. However, this is not necessarily possible for a circuit. The string $f_{v_i} \in \{0,1\}^\lambda$ must be hard-wired in a circuit using $O(\lambda / \log \lambda)$ gates. Then, the bits of $\mathrm{Enc}(f_{v_i})$ specified by $z_{S_i}$ must be computed from such embedded gates using $O(\lambda / \log \lambda)$ gates. This does not seem to be possible, as each bit of $\mathrm{Enc}(f_{v_i})$ depends on almost all bits of $f_{v_i}$ in order for Enc to be a good error-correcting code; thus, just reading such bits amounts to $O(\lambda)$ gates, which is too large. One may be tempted to let Enc be the identity function to avoid such a computation, but this does not work because it significantly weakens the reconstruction property of NW. In general, there are two conflicting requirements on Enc(-):

1. We need to ensure that each bit of $\mathrm{Enc}(f)$ can be computed by reading a small number of bits from $f$.

2. Enc(-) must be a good list-decodable error-correcting code: Given a string that agrees with

Enc($f$) on a $(1/2+\epsilon)$-fraction of bits for a small parameter $\epsilon > 0$, $f$ must be identified with a small number of advice bits.

In other words, we need a "locally-encodable and list-decodable" error-correcting code. We observe that the derandomized hardness amplification theorem of Impagliazzo and Wigderson [IW97] can be seen as such a code: They showed that any function $f\colon \{0,1\}^{\log\lambda} \to \{0,1\}$ can be converted into a function $\widehat{f}\colon \{0,1\}^{O(\log\lambda)} \to \{0,1\}$ so that given a small circuit that computes $\widehat{f}$ on a $(1/2+\epsilon)$-fraction of inputs, there exists a small circuit that computes $f$ on almost all inputs. Letting Enc($f$) be $\widehat{f}$, we observe that the two properties are satisfied. This enables us to prove $\mathsf{NP}$-hardness of MKTP*.

Proving $\mathsf{NP}$-hardness of MCSP* requires an additional (and the last) ingredient. We need to ensure that $\widehat{f}$ can be computed by a circuit of size $O(\lambda/\log\lambda)$ for a random function $f\colon \{0,1\}^{\log\lambda} \to \{0,1\}$. To this end, we employ the theorem of Uhlig [Uhl74; Uhl92], which shows that for any function $f\colon \{0,1\}^{\log\lambda} \to \{0,1\}$, the $r$-wise direct product $f^r\colon \left(\{0,1\}^{\log\lambda}\right)^r \to \{0,1\}^r$ can be computed by a circuit of size $O(\lambda/\log\lambda)$ for $r = \lambda^{o(1/\log\log\lambda)}$. Since $\widehat{f}$ can be locally computed by using the output of $f$ on at most $r$ inputs, we obtain a circuit of size $O(\lambda/\log\lambda)$ that computes $\widehat{f}$.

## 3 Open Problems

We expect that our results open up several fruitful research directions, which we mention below.

**MCSP** A major open problem is to prove $\mathsf{NP}$-hardness of MCSP. Following the two-step paradigm of showing $\mathsf{NP}$-hardness of variants of MCSP, it suffices to present a reduction from MCSP* to MCSP. We expect that this task now becomes much easier than it was previously, because the reduction of Theorem 1.2 creates a large gap between the Yes instances and the No instances. It would be a promising research direction to reduce the approximate and partial variant of MCSP of Theorem 1.2 to MCSP.

It is also interesting to see if MCSP* for $\mathsf{AC}^0$ circuits is $\mathsf{NP}$-hard. In general, our results do not necessarily prove $\mathsf{NP}$-hardness of MCSP* for circuit classes with unbounded-fan-in gates. The reason is that an $O(1)$-fan-in circuit of size $s$ can be encoded as a binary string of $O(s\log s)$, whereas the binary encoding of an unbounded-fan-in circuit of size $s$ can be as large as $s^2$. It is an interesting open question to broaden the applicability of our results to unbounded-fan-in circuits.

**Heuristica** Another major open question is to improve the inapproximability factor $n^\epsilon$ of Theorem 1.1 to $1.01n$. As is demonstrated in Appendix A, this will have a significant consequence on Impagliazzo's five worlds; in particular, it excludes Heuristica. We expect that this requires a "non-black-box" reduction technique that exploits the efficiency of a program in the No case. Indeed, the literature on the limits of black-box reductions [FF93; BT06b; AGGM06; BB15; HW20] suggests that no nonadaptive randomized reduction can reduce $\mathsf{NP}$ to a black-box oracle that solves $\mathsf{NP}$ on average. Our proof techniques are algorithmic information theoretic and do not exploit the efficiency of programs in the No case; thus, we expect that our reduction techniques are subject to such a barrier.

We note, however, that excluding Heuristica is in principle achievable by a combination of current proof techniques: Hirahara [Hir18; Hir21] developed *non-black-box* but *relativizing* reduction techniques (thus subject to the relativization barrier of Impagliazzo [Imp11]; see also [HN21]); we developed *black-box* but *non-relativizing* reduction techniques. What remains is to combine these proof techniques to *simultaneously* overcome black-box and relativization barriers. One way to do this is to reduce the approximate problem of MINKT$^*$ (Theorem 1.2) to an approximate problem GapMINKT of MINKT, which is known to be in $\mathsf{P}$ in Heuristica [Hir18; Hir20a].

A less challenging but intriguing open problem is to improve the inapproximability factor $n^\epsilon$ to $n^{1-o(1)}$, which could be achievable by combining sophisticated PCP machinery with the proof techniques developed in this paper.

**Computational Learning Theory** Our proof techniques would have applications to the problem of learning parities, which is one of central research topics in computational learning theory (see, e.g., [KMV08; BGGS16] and references therein). Learning $k$-sparse parities by $k$-juntas is known to be $\mathsf{W}[1]$-hard [BGGS16]. We strongly conjecture that Theorem 1.1 can be improved to $\mathsf{NP}$-hardness of learning $s$-sparse parities by programs of size $s \cdot n^{1/(\log\log n)^{O(1)}}$, which would significantly improve [BGGS16]. In fact, the only missing piece is the following question on PCPs: Can the PCP system of Dinur, Harsha, and Kindler [DHK15] be made smooth in the sense that every coordinate of a proof is queried equally likely?[15] We expect that the proof techniques developed in, e.g., [BHPT20], that make PCP systems smooth can be used to affirmatively resolve this question.

More broadly, our results open up the possibility of showing $\mathsf{NP}$-hardness of PAC learning of various concept classes by small programs. There are many non-proper PAC learner in the literature (e.g., [KMV08; LMN93; CIKK16], to name a few). Can we give evidence that such PAC learners cannot produce small programs, using our proof techniques of showing $\mathsf{NP}$-hardness?

**Organization** The remainder of this paper is organized as follows. In Section 4, we review the notion of Kolmogorov complexity and secret sharing scheme. Section 5 proves $\mathsf{NP}$-hardness of the Collective Minimum Weight Monotone Satisfying Assignment problem. Section 6 proves algorithmic information extraction lemmas. Sections 7 and 8 complete the proofs of Theorems 1.1 and 1.2, respectively. Appendix A presents the connection to excluding Heuristica. Appendix B extends Ko's relativization barrier to randomized reductions. In Appendix C, we mention a folklore result of $\mathsf{NP}$-hardness of learning size-$s$ circuits by size-$s$ circuits.

## 4 Preliminaries

We review the notion of Kolmogorov complexity and secret sharing scheme.

### 4.1 Kolmogorov Complexity

In order to show $\mathsf{NP}$-hardness of MKTP$^*$, we define time-bounded Kolmogorov complexity in such a way that it is meaningful even for sublinear-time bounds, following [ABKMR06]: A program is given random access to the data hard-wired in the program. For a string $x \in \{0,1\}^n$ and an integer $i \in [n+1]$, let $x_i$ denote the $i$-th bit of $x$ if $i \le |x|$ and $x_i := \bot$ otherwise. A program for describing $x$ is required to compute $x_i$ for each input $i$. In this way, we may define the KT-complexity of a string $x$, which is polynomially related to the circuit complexity of $x$, i.e., the minimum size of

[15] In fact, it suffices to have $|\Psi(x)| \ll n^{1/(\log\log n)^{O(1)}}$ in the notation of Theorem 5.2. We suspect that the original construction of [DHK15] might already satisfy this weak condition.

circuits that compute $x_i$ on input $i$. More formally, we fix an efficient universal Turing machine $U$. The time-bounded Kolmogorov complexity is defined as follows.

**Definition 4.1** (Time-bounded Kolmogorov complexity)**.** *For strings $x, y \in \{0,1\}^*$, a time bound $t \in \mathbb{N} \cup \{\infty\}$, and an oracle $A$, the $A$-oracle $t$-time-bounded Kolmogorov complexity of $x$ given $y$ is defined as*

$$\mathrm{K}^{t,A}(x \mid y) := \min\Big\{ |d| \;\Big|\; U^{A,d,y} \text{ outputs } x_i \text{ on input } i \in [|x|+1] \text{ in time } t \Big\}.$$

*Here, $U^{A,d,y}$ indicates that $U$ is given oracle access to $A$ and random access to $d$ and $y$. We omit the superscript $A$ if $A = \emptyset$, the superscript $t$ if $t = \infty$, and "$\mid y$" if $y$ is the empty string. We define*

$$\mathrm{KT}(x) := \min\big\{ s + t \;\big|\; \mathrm{K}^t(x) \le s \big\}.$$

A simple counting argument implies the following basic fact of Kolmogorov-randomness.

**Fact 4.2.** *For any integer $s \ge 1$ and any string $y \in \{0,1\}^*$, the number of strings $x \in \{0,1\}^*$ such that $\mathrm{K}(x \mid y) < s$ is less than $2^s$.*

*Proof.* The number of programs of length less than $s$ is at most $\sum_{i=0}^{s-1} 2^i < 2^s$. □

## 4.2 Secret Sharing Scheme

A *secret sharing scheme*, independently introduced by Shamir [Sha79] and Blakley [Bla79], enables sharing a secret among $n$ parties so that an "authorized" set of parties can reconstruct the secret, whereas any unauthorized set of parties obtains no information about the secret. Formally, whether a set of parties is authorized or not is represented by an access structure.

**Definition 4.3** (Access Structure)**.** *An* access structure *$\mathcal{A} \subseteq 2^{[n]}$ is a "monotone" collection of subsets of $[n]$; that is, for every $T \supseteq S \in \mathcal{A}$, we have $T \in \mathcal{A}$. A monotone function $f\colon \{0,1\}^n \to \{0,1\}$ is said to* represent *$\mathcal{A}$ if $f(\chi_T) = 1 \Leftrightarrow T \in \mathcal{A}$ for every subset $T \subseteq [n]$, where $\chi_T \in \{0,1\}^n$ is the characteristic vector of $T$. A subset $T \in \mathcal{A}$ is said to be* authorized *(with respect to $\mathcal{A}$); otherwise, it is said to be* unauthorized*.*

A secret sharing scheme consists of a randomized "sharing" algorithm Share and a deterministic "reconstruction" algorithm Rec. Given a secret $b \in \{0,1\}$, the algorithm $\mathrm{Share}(b)$ outputs $n$ shares $s_1, \ldots, s_n$; the $i$-th share is given to the $i$-th party.[16] A secret sharing scheme must satisfies two properties, correctness and privacy:

**Definition 4.4** (Secret Sharing Scheme [Bei11])**.** *A* secret sharing scheme *for $\mathcal{A} \subseteq 2^{[n]}$ is a pair* (Share, Rec) *of a randomized algorithm* Share *and a deterministic algorithm* Rec *with the following properties:*

**Correctness:** *For every authorized set $T \in \mathcal{A}$ and for every bit $b \in \{0,1\}$, the output of* $\mathrm{Share}(b)$ *is a sequence $(s_1, \ldots, s_n)$ of $n$ strings that satisfies* $\mathrm{Rec}(T, s_T) = b$ *with probability* 1 *over the internal randomness of* $\mathrm{Share}(b)$*.*

[16] Although a secret sharing scheme can share a *string* instead of *one bit b* in general, this is sufficient for our purpose.

**Privacy:** *For every unauthorized set $T \notin \mathcal{A}$ and for every random variable $b$ on $\{0,1\}$, the random variables $b$ and* $\mathrm{Share}(b)_T$ *are statistically independent.*

It is known that there exists an "efficient" secret sharing scheme for the access structures represented by monotone formulas.

**Lemma 4.5** (Ito, Saio, and Nishizeki [ISN93] and Benaloh and Leichter [BL88])**.** *Let $\mathcal{A} = \{\mathcal{A}_\varphi\}_\varphi$ be the family of access structures $\mathcal{A}_\varphi$ represented by monotone formulas $\varphi$. Then, there exists a pair of a randomized polynomial-time algorithm* Share *and a deterministic polynomial-time algorithm* Rec *such that for every monotone formula $\varphi$, the pair* $(\mathrm{Share}(\varphi, \text{-}), \mathrm{Rec}(\varphi, \text{-}))$ *is a secret sharing scheme for the access structure $\mathcal{A}_\varphi$. Moreover, the length $|s_i|$ of each share $s_i$ is at most the number $|\varphi|$ of the literals in the formula $\varphi$. For each fixed $\varphi$ and $T$, the reconstruction algorithm* $\mathrm{Rec}(\varphi, T, \text{-})$ *computes a linear function of $s_T$ over* $\mathrm{GF}(2)$*.*

We refer the reader to the survey of Beimel [Bei11] for a broad overview of secret sharing scheme.

## 5 Collective Minimum Monotone Satisfying Assignment Problem

We introduce a variant of the Minimum Monotone Satisfying Assignment problem [ABMP01; DS04], which we call the *Collective Minimum (Weight) Monotone Satisfying Assignment* (CMMSA) problem. The original problem asks to compute the minimum Hamming weight of an assignment that satisfies a given monotone formula. Here, we generalize the problem to a *weighted* version in which each variable has its own weight and a *collective* version in which the goal is to find the minimum weight of an assignment that satisfies as many as formulas in a given collection of monotone formulas.

To formally define the problem, we prepare some notation. For a monotone formula $\varphi$ on $n$ variables and a function $w \colon [n] \to \mathbb{N}$, the *weight* of an assignment $\alpha \colon [n] \to \{0,1\}$ with respect to $w$ is defined to be $\sum_{i=1}^{n} \alpha(i) \cdot w(i)$ and is denoted by $w(\alpha)$. Let $\varphi(\alpha) \in \{0,1\}$ denote the output of $\varphi$ when the variables are assigned by $\alpha$. For a subset $T \subseteq [n]$, let $w(T) := \sum_{i \in T} w(i)$. Now, we present the formal definition of CMMSA.

**Definition 5.1.** *The* Collective Minimum (Weight) Satisfying Assignment problem (CMMSA) with gap $g \in \mathbb{N}$ and soundness $\epsilon > 0$ *is the following problem. The input consists of a collection $\Phi = \{\varphi_1, \dots, \varphi_m\}$ of monotone formulas over the set $[n]$ of input variables, the weight function $w \colon [n] \to \mathbb{N}$ represented in unary, and a threshold parameter $s \in \mathbb{N}$. The task is to distinguish the following two cases.*

**Yes:** *There exists an assignment $\alpha \colon [n] \to \{0,1\}$ such that*

$$w(\alpha) \le s \quad \textit{and} \quad \Pr_{\varphi \sim \Phi}[\varphi(\alpha) = 1] = 1.$$

**No:** *For every assignment $\alpha \colon [n] \to \{0,1\}$, if $w(\alpha) \le g \cdot s$, then*

$$\Pr_{\varphi \sim \Phi}[\varphi(\alpha) = 1] < \epsilon.$$

*The* degree *of $\Phi$ is defined to be $\max_{\varphi \in \Phi} |\varphi|$, where $|\varphi|$ denotes the number of the literals in the formula $\varphi$. The size of an instance of* CMMSA *is measured by the number $n$ of input variables.*

Dinur and Safra [DS04] showed NP-hardness of the MMSA problem. We apply their reduction to CMMSA.

**Theorem 5.2.** *For any constant $\beta > 0$, there exists a constant $\alpha > 0$ such that for every parameter $\Delta\colon \mathbb{N} \to \mathbb{N}$ such that $\omega(1) \le \Delta(n) \le 2^{(\log n)^{1-\beta}}$ for all large $n \in \mathbb{N}$, it is* NP*-hard under polynomial-time many-one reductions to compute* CMMSA *with gap $\Delta(n)^{\alpha}$, degree $\Delta(n)$, and soundness $\Delta(n)^{-\alpha}$ on a collection $\Phi$ of monotone DNF formulas over $n$ variables.*

We will use this result for $\Delta(n) := (\log n)^{1/2}$. For our applications, it is important to maximize the gap $\Delta(n)^{\alpha}$ for a fixed degree $\Delta(n)$ because in the NP-hardness reductions of learning, the input length of a target function is determined by the degree $\Delta(n)$, and the inapproximability factor is determined by the gap $\Delta(n)^{\alpha}$. To obtain the large gap in Theorem 5.2, we employ PCP systems from the literature on the *Sliding Scale Conjecture*. The Sliding Scale Conjecture [BGLR94] states that for every parameter $\delta$ such that $\frac{1}{\mathsf{poly}(n)} \le \delta < 1$, every language in NP has a PCP verifier that tosses $O(\log n)$ random coins, makes $O(1)$ queries into a proof over an alphabet $\Sigma$ of size $\mathsf{poly}(1/\delta)$, has perfect completeness, and soundness error $\delta$. This conjecture was resolved for any sufficiently large soundness error $\delta \ge 2^{-(\log n)^{1-\beta}}$ for a constant $\beta > 0$.

**Lemma 5.3** ([DFKRS11; DHK15])**.** *Let $\beta > 0$ be a constant and $\delta\colon \mathbb{N} \to [0,1]$ be a function such that $\delta(n) \ge 2^{-(\log n)^{1-\beta}}$ for all large $n \in \mathbb{N}$. Then, every language $L \in$ NP admits an $O(1/\beta)$-query PCP system over an alphabet size $\mathsf{poly}(1/\delta(n))$ with randomness complexity $O(\log n)$, soundness error $\delta$, and perfect completeness on inputs of length $n$.*

This was originally proved by Dinur, Fischer, Kindler, Raz, and Safra [DFKRS11]. Although the main result of [DFKRS11] is stated only for $\delta(n) = 2^{-(\log n)^{1-\beta}}$ for an arbitrary constant $\beta \in (0,1)$, the subsequent work of Dinur, Harsha, and Kindler [DHK15] gave a streamlined proof of [DFKRS11], from which a PCP system with larger soundness error $\delta(n) \ge 2^{-(\log n)^{1-\beta}}$ can be obtained.

Using this PCP system, we present a proof of Theorem 5.2.

*Proof of Theorem 5.2.* The PCP theorem of Lemma 5.3 can be stated in terms of MaxCSP as follows: Let $\Psi = \{C_1, \ldots, C_m\}$ be the set of constraints over $n$ variables on the alphabet $\Sigma$. Here, for any internal randomness $j \in \{0,1\}^{O(\log n)}$ of the PCP verifier of Lemma 5.3, there is a constraint $C_j$. Each constraint $C_j$ depends on exactly $D = O(1/\beta)$ variables. The size of the alphabet $\Sigma$ is at most $\mathsf{poly}(1/\delta)$. Let $C_j^{-1}(1)$ denote the set of assignments to the variables in $C_j$ that cause $C_j$ to accept. Here, an assignment $r$ to the variables in $C_j$ is a function $r\colon \mathrm{dom}(r) \to \Sigma$, where $\mathrm{dom}(r) \subseteq [n]$ denotes the set of variables in $C_j$.

Given the MaxCSP instance $\Psi$ over $\Sigma$, we reduce it to an instance $(\Phi, w\colon [n] \times \Sigma \to \mathbb{N}, s \in \mathbb{N})$ of CMMSA as follows: Each variable of $\Phi$ is indexed by $(x,a) \in [n] \times \Sigma$ and is denoted by $L_{x,a}$. Informally, $L_{x,a} = 1$ indicates that the variable $x$ in the original CSP instance $\Psi$ is assigned to $a \in \Sigma$. For each $j \in [m]$, construct a monotone DNF formula $\varphi_j$ defined as

$$\varphi_j(L) := \bigvee_{r \in C_j^{-1}(1)} \bigwedge_{x \in \mathrm{dom}(r)} L_{x,r(x)}.$$

We define the weight $w(x,a)$ of $L_{x,a}$ to be $|\Psi(x)|$, where $\Psi(x)$ is the set of indices $j \in [n]$ such that $C_j$ contains $x$ as a variable. We define $s := mD$.

We prove the correctness of the reduction. Assume that the CSP instance $\Psi$ is satisfied by an assignment $\alpha\colon [n] \to \Sigma$. Then, we set $L_{x,\alpha(x)} := 1$ and $L_{x,y} := 0$ for every $y \in \Sigma \setminus \{\alpha(x)\}$. Since $w(x,\alpha(x)) = |\Psi(x)|$, the weight of the assignment $L\colon [n] \times \Sigma \to \{0,1\}$ is

$$w(L) = \sum_{x \in [n]} |\Psi(x)| = |\{(x,j) \in [n] \times [m] \mid j \in \Psi(x)\}| = mD.$$

Here, we used that each constraint $C_j$ depends on exactly $D$ variables. For every $j \in [m]$, we have $C_j(\alpha) = 1$; thus, $\varphi_j(L) = 1$. This implies that $(\Phi, w, s)$ is a Yes instance of CMMSA.

Next, assume that any assignment to $\Psi$ can satisfy at most a $\delta$-fraction of constraints in $\Psi$, where $\delta$ is a soundness parameter of Lemma 5.3 chosen later depending on a given parameter $\Delta$. Assume that there exists an assignment $L\colon [n] \times \Sigma \to \{0,1\}$ such that $w(L) = g \cdot mD$ and

$$\Pr_{j \sim [m]}[\varphi_j(L) = 1] \geq \epsilon, \tag{3}$$

where $\epsilon > 0$ is a parameter to be chosen later. We claim that $g$ must be large. For each variable $x \in [n]$ of $\Psi$, let $A(x) := \{a \in \Sigma \mid L_{x,a} = 1\}$. By the definition of the weight function $w(\text{-})$, we have

$$w(L) = \sum_{x} |\Psi(x)| \cdot |A(x)| \leq g \cdot mD. \tag{4}$$

Consider the distribution $\mathcal{D}$ on the set $[n]$ of variables defined by the following sampling procedure: Choose $C_j \sim \Psi$ uniformly and randomly, choose a variable $x$ from the variables of $C_j$ uniformly and randomly, and output $x$. Each variable $x$ is chosen with probability $|\Psi(x)|/mD$; thus, Eq. (4) is equivalent to

$$\mathop{\mathbb{E}}_{x \sim \mathcal{D}}[|A(x)|] \leq g.$$

Therefore, we obtain

$$\begin{aligned}
\frac{\epsilon}{2D} &\geq \frac{\epsilon}{2gD} \cdot \mathop{\mathbb{E}}_{x \sim \mathcal{D}}[|A(x)|] \\
&\geq \Pr_{x \sim \mathcal{D}}\left[|A(x)| \geq \frac{2gD}{\epsilon}\right] && \text{(by Markov's inequality)} \\
&\geq \Pr_{C \sim \Psi}\left[|A(x)| \geq \frac{2gD}{\epsilon} \text{ for some variable } x \text{ in } C\right] \cdot \frac{1}{D}. && \text{(since } C \text{ contains } D \text{ variables)}
\end{aligned}$$

Combining this inequality with Eq. (3), we get

$$\Pr_{C \sim \Psi}\left[\exists r \in C^{-1}(1), \forall x \in \mathrm{dom}(r),\ |A(x)| \leq \frac{2gD}{\epsilon} \text{ and } r(x) \in A(x)\right] \geq \epsilon - \frac{\epsilon}{2} = \frac{\epsilon}{2}.$$

Now, we construct a random assignment $\alpha\colon [n] \to \Sigma$ as follows: For each $x \in [n]$, pick $a \sim A(x) \subseteq \Sigma$ uniformly and randomly and define $\alpha(x) := a$. Under the event that $r \in C^{-1}(1)$, $|A(x)| \leq \frac{2gD}{\epsilon}$, and $r(x) \in A(x)$ for every $x \in \mathrm{dom}(r)$, we have $C(\alpha) = 1$ if $\alpha(x) = r(x)$ for every $x \in \mathrm{dom}(r)$, which happens with probability at least $\left(\frac{\epsilon}{2gD}\right)^D$. It follows that

$$\Pr_{\substack{C \sim \Psi \\ \alpha}}[C(\alpha) = 1] \geq \frac{\epsilon}{2} \cdot \left(\frac{\epsilon}{2gD}\right)^D.$$

By an averaging argument, there exists an assignment $\alpha\colon [n] \to \Sigma$ that satisfies a $\frac{\epsilon}{2} \cdot \left(\frac{\epsilon}{2gD}\right)^D$-fraction of constraints in $\Psi$. By the assumption on $\Psi$, we have

$$\frac{\epsilon}{2} \cdot \left(\frac{\epsilon}{2gD}\right)^D \leq \delta,$$

which implies that

$$g \geq \left(\frac{\epsilon}{2}\right)^{1+\frac{1}{D}} \cdot \delta^{-\frac{1}{D}} \cdot \frac{1}{D} \geq \Omega\left(\epsilon^2 \cdot \delta^{-\frac{1}{D}}\right).$$

Setting $\epsilon := \delta^{\frac{1}{4D}}$, we obtain

$$g \geq \Omega\left(\delta^{-\frac{1}{2D}}\right).$$

The number of the literals in $\varphi_j \in \mathit{\Phi}$ is at most $|C_j^{-1}(1)| \cdot D \leq |\Sigma|^D \cdot D \leq \delta^{-O(D)}$. Given a parameter $\Delta$, we choose $\delta := \Delta^{-\Omega(1/D)}$ so that the degree of $\mathit{\Phi}$ is at most $\Delta$. Then, the gap $g$ is at least $\Omega\left(\delta^{-\frac{1}{2D}}\right) \geq \Delta^{\Omega(1/D^2)}$. Moreover, the soundness $\epsilon$ is at least $\delta^{\frac{1}{4D}} \geq \Delta^{-\Omega(1/D^2)}$. $\square$

**Remark 5.4.** Dinur and Safra [DS04] showed that the unweighted version of CMMSA (in which $w \equiv 1$) is NP-hard. The output of their reduction is a depth-3 formula $\varphi := \bigwedge \mathit{\Phi}$, where $\mathit{\Phi}$ is the collection of monotone DNF formulas in the proof of Theorem 5.2. They made the instance unweighted by replacing each variable $L_{x,a}$ with $\bigwedge_{i \in [w(x,a)]} T_{x,a}^i$, where $T_{x,a}^i$ is a fresh variable for each $i$. Note that $L_{x,a}$ can be set to 1 if and only if $T_{x,a}^i = 1$ for every $i \in [w(x,a)]$. In this way, one can transform a weighted version into a unweighted version. However, this transformation potentially increases the degree of $\Phi$; thus, we chose to present the reduction for a weighted version.

For the proof of Theorem 1.1, we use NP-hardness of approximating MMSA.

**Lemma 5.5** ([DS04; DHK15]; see also [Hir22])**.** *It is* NP*-hard to compute* CMMSA *with gap* $g = n^{1/(\log\log n)^{O(1)}}$ *on inputs* $\mathit{\Phi} = \{\varphi\}$*, where* $\varphi$ *is a depth-3 monotone formula over* $n$ *variables.*

*Proof Sketch.* Dinur, Harsha, and Kindler [DHK15] showed a weak variant of the Sliding Scale Conjecture in which the number of queries is relaxed to be $D = (\log\log n)^{O(1)}$ and for the soundness error $\delta(n) = 1/\mathsf{poly}(n)$. Combining this with the reduction of [DS04], we obtain NP-hardness of approximating MMSA to within a factor of $n^{1/(\log\log n)^{O(1)}}$. $\square$

## 6 Algorithmic Information Extraction Lemmas

In this section, we present algorithmic information extraction lemmas. In the case of the $k$-wise direct product generator, the lemma can be formally stated as follows.

**Lemma 6.1.** *Let* $a, k, \epsilon^{-1} \in \mathbb{N}$*,* $f_1, \ldots, f_m \in \{0,1\}^\lambda$*, and* $D\colon \{0,1\}^a \times \left(\{0,1\}^{\lambda k + k}\right)^m \to \{0,1\}$ *be a function. Then, there exists a set* $B \subseteq [m]$ *such that*

$$\mathrm{K}^D(f_B) \leq |B| \cdot (mk + a + O(\log(m\lambda k a/\epsilon)))$$

*and for every* $\alpha \in \{0,1\}^a$*,*

$$\left|\Pr[D(\alpha, X_1, \ldots, X_m) = 1] - \Pr\left[D(\alpha, X'_1, \ldots, X'_m) = 1\right]\right| \leq \epsilon.$$

*Here,* $X_i$ *is the random variable identical to* $\mathrm{DP}_k(f_i; z_i)$ *for a random choice of* $z_i \sim \{0,1\}^{\lambda k}$*.* $X'_i$ *is identical to* $X_i$ *if* $i \in B$ *and to the uniform distribution if* $i \in [m] \setminus B$*.*

This lemma shows that one can extract the set $B$ of indices such that $D$ "knows" $f_i$ for all $i \in B$ and $D$ "does not know" $f_i$ for all $i \notin B$. The first condition ensures that the set $B$ is small (by a calculation similar to Eq. (2)). The second condition shows that $D$ cannot distinguish the output distribution of $\mathrm{DP}_k(f_i; \text{-})$ from the uniform distribution for every $i \notin B$. For some technical reason, we supply an advice string $\alpha$ to $D$; this will be used in order to provide the information on a secret sharing scheme without affecting the choice of $B$.

For the purpose of showing $\mathsf{NP}$-hardness of MCSP*, we use the Nisan–Wigderson pseudorandom generator construction. We will state an algorithmic extraction lemma for the Nisan–Wigderson generator and prove Lemma 6.1 as a special case. We recall the notion of design and the definition of the Nisan–Wigderson generator.

**Proposition 6.2** ([NW94; Tre01]). *For any sufficiently large parameters $\ell, m, \rho \in \mathbb{N}$ with $m \le 2^\ell$, there exists a "design" $S_1, \dots, S_m \subseteq [d]$ such that for every $i \in [m]$,*

1. *$|S_i| = \ell$, $d = O(\exp(\ell/\rho) \cdot \ell^2/\rho)$, and*
2. *$|S_i \cap S_j| \le \rho$ for every $j \in [m] \setminus \{i\}$*

*Moreover, such a family can be constructed in time $\mathsf{poly}(2^d, m)$.*

**Definition 6.3** (The Nisan–Wigderson pseudorandom generator construction [NW94]). *Let $\mathcal{S} = (S_1, \dots, S_m)$ be a family of $\ell$-sized subsets of $[d]$. We define a function*

$$\mathrm{NW}_{\mathcal{S}} \colon \{0,1\}^{2^\ell} \times \{0,1\}^d \to \{0,1\}^m$$

*as*

$$\mathrm{NW}_{\mathcal{S}}(f; z) := (f(z_{S_1}), \dots, f(z_{S_m})) \in \{0,1\}^m,$$

*where we identify a string $f \in \{0,1\}^{2^\ell}$ with a Boolean function $f \colon \{0,1\}^\ell \to \{0,1\}$ and $z_{S_i} \in \{0,1\}^\ell$ denotes the string obtained by concatenating all the bits of $z$ indexed by $S_i$.*

Nisan and Wigderson [NW94] showed that if any small circuit fails to compute $f$ on a $(1/2 - o(1))$-fraction of inputs, then $\mathrm{NW}_{\mathcal{S}}(f; z)$ is a pseudorandom generator secure against small circuits. To state this reconstruction property formally, we recall an average-case version of Kolmogorov complexity.

**Definition 6.4** ([ISW06; FLV06]). *For any $\delta < \frac{1}{2}$, any string $x \in \{0,1\}^*$, and any oracle $A \subseteq \{0,1\}^*$, let $\mathrm{K}^A_\delta(x)$ denote the minimum of $\mathrm{K}^A(y)$ over all strings $y \in \{0,1\}^{|x|}$ with Hamming distance at most $\delta|y|$ from $x$.*

Although reconstruction properties of pseudorandom generator constructions are usually given by randomized algorithms, it is possible to derandomize the algorithms by using a pseudorandom generator.

**Fact 6.5.** *Let $A \subseteq \{0,1\}^*$ be an oracle and $t \colon \mathbb{N} \to \mathbb{N}$ be a computable function. Then, there exists a pseudorandom generator*

$$G = \left\{ G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n \right\}_{n \in \mathbb{N}}$$

*secure against $A$-oracle $t(n)$-time programs of length $n$. The pseudorandom generator can be computed in finite steps given oracle access to $A$.*

*Proof Sketch.* We claim the existence of a pseudorandom generator $G_n\colon \{0,1\}^{s(n)} \to \{0,1\}^n$. Let $G_n$ be a uniformly random function. Fix any $A$-oracle $n$-input program $M^A$ of size $n$. Let $\epsilon := 1/n$. By Hoeffding's inequality, we have

$$\Pr_{G_n}\left[\left|\mathop{\mathbb{E}}_{z\sim\{0,1\}^{s(n)}}\left[M^A(G_n(z))\right] - \mathop{\mathbb{E}}_{w\sim\{0,1\}^n}\left[M^A(w)\right] \geq \epsilon\right|\right] \leq 2\exp\left(-2\epsilon^2 2^{s(n)}\right).$$

By taking a union bound over all programs of size $n$, the probability that there exists a program of size $n$ that can distinguish $G_n(\text{-})$ from the uniform distribution is at most $2^{O(n)} \cdot \exp\left(-2\epsilon^2 2^{s(n)}\right)$. For a sufficiently large $s(n) = O(\log(n/\epsilon))$, this probability is bounded above by $\frac{1}{2}$. In particular, there exists a pseudorandom generator $G_n\colon \{0,1\}^{s(n)} \to \{0,1\}^n$ secure against linear-sized $A$-oracle programs. Such a pseudorandom generator can be found by an exhaustive search. $\square$

We are now ready to state and prove the algorithmic information extraction lemma for the Nisan–Wigderson pseudorandom generator, which generalizes Lemma 6.1.

**Lemma 6.6.** *Let $a, \ell, d, m, \epsilon^{-1} \in \mathbb{N}$. Let $\mathcal{S} = (S_1, \dots, S_m)$ be a family of $\ell$-sized subsets of $[d]$. Let $f_1, \dots, f_m\colon \{0,1\}^\ell \to \{0,1\}$ and $D\colon \{0,1\}^a \times \{0,1\}^d \times \{0,1\}^m \to \{0,1\}$ be functions. Then, there exists a subset $B \subseteq [m]$ such that for every $i \in B$,*

$$\mathrm{K}^D_{\frac{1}{2}-\frac{\epsilon}{2m'}}(f_i) \leq \sum_{j\in[m]\setminus\{i\}} 2^{|S_i\cap S_j|} + a + O(\log(mad\ell'/\epsilon))$$

*and for every $\alpha \in \{0,1\}^a$,*

$$\left|\Pr[D(\alpha, Z, X_1, \dots, X_m) = 1] - \Pr\left[D(\alpha, Z, X_1', \dots, X_m') = 1\right]\right| \leq \epsilon.$$

*Here, $Z \sim \{0,1\}^d$, $X_i$ is the random variable identical to $f_i(Z_{S_i})$, and $X_i'$ is the random variable identical to $X_i$ if $i \in B$ and to a uniformly random bit $X_i' \sim \{0,1\}$ if $i \in [m] \setminus B$. We define $m'$ to be the maximum, over all $\alpha$, of the number of $i \in [m]$ such that $D(\alpha, \text{-}, \text{-})$ depends on $X_i$. We also define $\ell'$ to be $\max_{i\in[m]} \mathrm{K}^{2^{O(\ell)}}(f_i)$.*

*Proof.* For each $i \in [m]$, let $\theta_i := \sum_{j\in[m]\setminus\{i\}} 2^{|S_i\cap S_j|} + a + O(\log(mad\ell'/\epsilon))$ and let $\delta := \frac{1}{2} - \frac{\epsilon}{2m'}$. We define a set $B \subseteq [m]$ as

$$B := \left\{ i \in [m] \;\middle|\; \mathrm{K}^D_\delta(f_i) \leq \theta_i \right\}.$$

We prove that the function $D$ cannot distinguish the distribution $(\alpha, Z, X_1, \dots, X_m)$ from $(\alpha, Z, X_1', \dots, X_m')$. At a high level, this can be proved as follows: For any $i \in B$, the random variables $X_i$ and $X_i'$ are identical. For any $i \in [m] \setminus B$, the bit $X_i$ is one bit $f_i(Z_{S_i})$ of the output of the Nisan–Wigderson pseudorandom generator, which is indistinguishable from the uniformly random bit $X_i' \sim \{0,1\}$; otherwise, $f_i$ can be approximated by using $\theta_i$ bits of an advice string and the distinguisher $D$, which contradicts $i \notin B$, i.e., $\mathrm{K}^D_\delta(f_i) > \theta_i$.

To formalize this argument, we use a standard hybrid argument [Vad12]. Fix any $\alpha \in \{0,1\}^a$. Let $I \subseteq [m]$ be the set of indices $i$ such that $D(\alpha, \text{-}, \text{-})$ depends on $X_i$. Assume, toward a contradiction, that[17]

$$\Pr[D(\alpha, Z, X_1, \dots, X_m) = 1] - \Pr\left[D(\alpha, Z, X_1', \dots, X_m') = 1\right] \geq \epsilon.$$

[17]Without loss of generality, we can drop the absolute value in this inequality.

For each $i \in \{0, \dots, m\}$, let $H_i$ denote the random variable $(X_1, \dots, X_i, X'_{i+1}, \dots, X'_m)$. Since $H_m$ is identical to $(X_1, \dots, X_m)$ and $H_0$ is identical to $(X'_1, \dots, X'_m)$, we have

$$\Pr[D(\alpha, Z, H_m) = 1] - \Pr[D(\alpha, Z, H_0) = 1] \geq \epsilon.$$

Note that $H_{i-1}$ is identical to $H_i$ if $i \notin I$. Thus, there exists an index $i \in I$ such that

$$\Pr[D(\alpha, Z, H_i) = 1] - \Pr[D(\alpha, Z, H_{i-1}) = 1] \geq \frac{\epsilon}{m'},$$

where we used that $|I| \leq m'$. Moreover, $H_{i-1}$ is identical to $H_i$ if $i \in B$ by the definition of $X'_i$; thus, we have $i \notin B$. Our goal is to prove $i \in B$, which leads to a contradiction. By using Yao's next-bit predictor [Yao82], we obtain a randomized linear-sized $D$-oracle circuit $P^D$ such that

$$\Pr\left[P^D(\alpha, Z, X_1, \dots, X_{i-1}, X'_{i+1}, \dots, X'_m) = X_i\right] \geq \frac{1}{2} + \frac{\epsilon}{m'}.$$

Now, we pick $Z_{[d]\setminus S_i}$ randomly, uniform bits in $X'_{i+1}, \dots, X'_m$, and the internal randomness of $P^D$. Then, by an averaging argument, with probability at least $\frac{\epsilon}{2m'}$ over such a random choice, it holds that

$$\Pr_{Z_{S_i}}\left[P^D(\alpha, Z, X_1, \dots, X_{i-1}, X'_{i+1}, \dots, X'_m) = X_i\right] \geq \frac{1}{2} + \frac{\epsilon}{2m'} = 1 - \delta.$$

Here, the probability is taken over only $Z_{S_i} \sim \{0,1\}^{S_i}$ and the other random choices are fixed. Let $A$ be an advice function defined as follows: Each bit of the output of $A$ is indexed by the set $\mathcal{I}_i := \left\{(j, z_{S_i \cap S_j}) \in ([m] \setminus \{i\}) \times \{0,1\}^{|S_i \cap S_j|}\right\}$. For each $(j, z_{S_i \cap S_j}) \in \mathcal{I}_i$ and a given $z_{[d]\setminus S_i}$, we write down $f_j(z_{S_j})$ as one bit of the output of $A$. That is, $A(i, z, f_1, \dots, f_m) := \left(f_j(z_{S_j}) \mid (j, z_{S_i \cap S_j}) \in \mathcal{I}_i\right)$. Observe that $a'_i := |\mathcal{I}_i| = \sum_{j \in [m]\setminus\{i\}} 2^{|S_i \cap S_j|}$. Note that $(X_1, \dots, X_{i-1}, X'_{i+1}, \dots, X'_m)$ can be computed from $i$, $Z$, and the advice string $A(i, Z, f_1, \dots, f_m)$. Moreover, since $X_i = f_i(Z_{S_i})$, the circuit $P^D$ can be used to approximate $f_i$. Let $R^D$ be a randomized circuit that takes $\alpha, i, Z$, and an advice string $\beta \in \{0,1\}^{a'_i}$ and outputs the output bits of $P^D$ over all strings $Z_{S_i} \in \{0,1\}^{\ell}$. Then, we have

$$\Pr\left[\mathrm{dist}(f_i, R^D(\alpha, i, Z, A(i, Z, f_1, \dots, f_m))) \leq \delta\right] \geq \frac{\epsilon}{2m'}, \tag{5}$$

where $\mathrm{dist}(f, g)$ denotes the fractional Hamming distance between $f$ and $g$ and the probability is taken over $Z$ and the internal randomness of $R^D$. Now, we derandomize this random choice using a pseudorandom generator $G\colon \{0,1\}^{O(\log N)} \to \{0,1\}^N$ secure against $D$-oracle linear-sized programs. Note that the condition that $f_i$ is $\delta$-close to $R^D(\alpha, i, Z, A(i, Z, f_1, \dots, f_m))$ can be checked by a $D$-oracle program of size $N = \mathsf{poly}(\ell', m, d, a, 1/\epsilon)$ because we assumed that each $f_i$ has Kolmogorov complexity at most $\ell'$. Using the pseudorandom generator $G$ of Fact 6.5, there exists a seed of length $O(\log N)$ such that the randomness used in Eq. (5) can be replaced while the probability remains to be at least $\frac{\epsilon}{2m'} - \frac{1}{N} > 0$. In particular, there exists a seed of length $O(\log N)$ such that the reconstruction procedure $R^D$ prints a string $\delta$-close to $f_i$. Now, we are ready to describe a $D$-oracle program $M^D$ that approximately describes $f_i$: The program $M^D$ takes a seed of $G$, $i \in [m]$, $\alpha \in \{0,1\}^a$, an advice string $\beta \in \{0,1\}^{a'_i}$ (which is supposed to be $A(i, Z, f_1, \dots, f_m)$), generates $Z$ and the internal randomness of $R^D$ by using $G$, and outputs $R^D(\alpha, i, Z, \beta)$. Since this program approximates $f_i$, we obtain

$$\mathrm{K}^f_\delta(f_i) \leq a'_i + a + O(\log m a d \ell'/\epsilon) \leq \theta_i,$$

which implies $i \in B$. However, this contradicts the fact that $i \notin B$. □

The version of an algorithmic information extraction lemma for the $k$-wise direct product generator is an immediate corollary of Lemma 6.6.

*Proof of Lemma 6.1.* We apply Lemma 6.6 to the Hadamard encoding of $f_1, \dots, f_m$ and a disjoint family $\mathcal{S} = (S_1, \dots, S_{mk})$. Let $\mathcal{S} = (S_1, \dots, S_{mk})$ be a disjoint family of $\lambda$-sized subsets of $[\lambda km]$. For each $i \in [m]$, let $\widehat{f}_i \colon \{0,1\}^\lambda \to \{0,1\}$ denote the function that takes $x \in \{0,1\}^\lambda$ and outputs the inner product $\langle x, f_i \rangle := \sum_{j=1}^{\lambda} x_j f_{ij} \mod 2$. Let $f'_{(i-1)\cdot k+j} := \widehat{f}_i$ for any $j \in [k]$. By applying Lemma 6.6 to $f'_1, \dots, f'_{mk}$ and $S_1, \dots, S_{mk}$, we obtain a set $B \subseteq [m]$ such that for every $i \in B$,

$$\mathrm{K}^D_{\frac{1}{2}-\frac{\epsilon}{2mk}}(\widehat{f}_i) \le m - 1 + a + O(\log(m\lambda ak/\epsilon)),$$

where we used that $S_i \cap S_j = \emptyset$ for every distinct pair $(i,j)$ and that $\ell' = \max_i \mathrm{K}^{2^{O(\lambda)}}(\widehat{f}_i) \le O(\lambda)$. Using the list-decoding algorithm of the Hadamard code [GL89], we obtain

$$\mathrm{K}^D(f_i) \le m + a + O(\log(m\lambda ak/\epsilon)).$$

Since $f_B$ can be described by programs of length $\mathrm{K}^D(f_i)$ that print $f_i$ for each $i \in B$, we have

$$\mathrm{K}^D(f_B) \le \sum_{i \in B} \mathrm{K}^D(f_i) + O(\log \lambda) \le |B| \cdot (m + a + O(\log(m\lambda ak/\epsilon))).$$

This completes the proof of the first property.

Let $z = (z_1, \dots, z_m) \in \left(\{0,1\}^{\lambda k}\right)^m$ be a seed. The second property is immediate from Lemma 6.6 because $X_{(i-1)\cdot k+j}$ of Lemma 6.6 is identical to the $j$-th bit of $f_i \cdot z_i \in \{0,1\}^k$ for every $i \in [m]$ and every $j \in [k]$ (Recall that $\mathrm{DP}_k(f_i; z_i) = (z_i, f_i \cdot z_i)$). $\square$

## 7 NP-Hardness of Learning Programs

Using the tools developed in the previous two sections, we now present a proof of NP-hardness of learning programs.

We formally define the problem of learning programs. For a distribution $\mathcal{E}$, we say that $\mathcal{E}$ is *represented* by a circuit $S$ if the output $S(r)$ of the circuit $S$ over a uniformly random string $r \sim \{0,1\}^{|S|}$ is identical to the distribution $\mathcal{E}$.

**Definition 7.1** (Learning Programs)**.** *Let $t \colon \mathbb{N} \to \mathbb{N}$, $g \colon \mathbb{N} \to \mathbb{N}$, and $\epsilon \colon \mathbb{N} \to (0,1]$ be functions. We define* $\mathrm{Gap}_{t,g,\epsilon}\mathrm{Learn}$ *to be the following promise problem: The input consists of a size parameter $s \ge n^{\Omega(1)}$ represented in unary and a distribution $\mathcal{E}$ represented by a circuit such that $\mathrm{supp}(\mathcal{E}) \subseteq \{0,1\}^n \times \{0,1\}$ for some $n \in \mathbb{N}$.*[18] *The task is to distinguish the following two cases:*

**Yes:** *There exists a $t(n)$-time program $M$ of size $s$ such that*

$$\Pr_{(x,b)\sim\mathcal{E}}[M(x) = b] = 1.$$

**No:** *For any program $M$ of size $s \cdot g(n)$,*

$$\Pr_{(x,b)\sim\mathcal{E}}[M(x) = b] < \frac{1}{2} + \frac{\epsilon(n)}{2}.$$

[18]The lower bound on $s$ is to make sure that the relativization barrier, which is applicable to additive approximation versions, of Proposition B.1 holds. Note that this restriction makes NP-hardness stronger.

**Theorem 7.2** (Restatement of Theorem 1.1)**.** *For every constant $\delta > 0$, there exist a polynomial $t$ and a function $g(n) = n^{1/(\log\log n)^{O(1)}}$ such that* $\mathrm{Gap}_{t,g,\epsilon}\mathrm{Learn}$ *is* $\mathsf{NP}$*-hard under randomized polynomial-time many-one reductions, where $\epsilon(n) := 2^{-n^{1-\delta}}$. Moreover, in the* Yes *case, there exists a program $M$ of size $s$ that computes some linear function over $x \in \{0,1\}^n$ that outputs $b$ for every $(x,b) \in \mathrm{supp}(\mathcal{E})$.*

*Proof.* By Lemma 5.5, it suffices to present a randomized reduction $R$ from the MMSA problem to $\mathrm{Gap}_{t,g,\epsilon}\mathrm{Learn}$. Let $(\varphi, \theta)$ be an instance of MMSA, where $\varphi$ is a monotone formula over $n$ variables and $\theta \in \mathbb{N}$ is a threshold parameter. The reduction $R$ picks $f_i \sim \{0,1\}^\lambda$ randomly for each $i \in [n]$. Then, $R$ outputs a distribution $\mathcal{E} = \mathcal{E}(f_1, \dots, f_n)$ from which one can sample $(x,b)$ as follows: Pick a secret $b \sim \{0,1\}$ randomly. Using Lemma 4.5, share the secret $b$ among $n$ parties; that is, let $(s_1, \dots, s_n) := \mathrm{Share}(\varphi, b)$, where each $s_i$ is the share given to the $i$-th party. We may assume without loss of generality that the length of $s_i$ is exactly equal to $k$ for every $i \in [n]$. The input $x$ is defined to be

$$(\mathrm{DP}_k(f_1; z_1) \oplus s'_1, \dots, \mathrm{DP}_k(f_n; z_n) \oplus s'_n) = ((z_1, f_1 \cdot z_1 \oplus s_1), \dots, (z_n, f_n \cdot z_n \oplus s_n)),$$

where $s'_i := (0^{\lambda k}, s_i)$. Here, we define $\lambda := \max\{(2nk)^{2/\delta}, |\varphi|^2\}$. Note that the length of the input $x$ is $n \cdot (\lambda k + k) \le 2nk\lambda$; thus, we have $|x| \le \lambda^{1+\delta/2}$.

We prove the completeness of the reduction $R$. Assume that there exists an assignment $\alpha\colon [n] \to \{0,1\}$ such that $\sum_{i=1}^n \alpha(i) \le \theta$. Let $T := \{i \in [n] \mid \alpha(i) = 1\}$. We claim that there exists an efficient program $M$ of size $(1 + o(1)) \cdot \theta\lambda$ that computes $b$ on input $x$ for every $(x,b) \in \mathrm{supp}(\mathcal{E})$. The program $M$ takes $\varphi$, $T$, and $\{f_i \mid i \in T\}$ as hard-wired input and $x = (\xi_1, \dots, \xi_n)$ as input, defines $(z_i, \xi'_i) := \xi_i$, computes $s_i := (f_i \cdot z_i) \oplus \xi'_i$ for each $i \in T$, and outputs $b = \mathrm{Rec}(\varphi, T, s_T)$. Since $T$ is an authorized set with respect to the access structure $\mathcal{A}_\varphi$ represented by $\varphi$, the correctness of this algorithm follows from the correctness of the secret sharing scheme. The running time of $M$ is clearly bounded by some polynomial $t(n)$ in $n$. The size of $M$ is at most

$$O(|\varphi| \log |\varphi|) + |T| \cdot O(\log n) + \sum_{i \in T} |f_i| \le (1 + o(1)) \cdot \lambda\theta,$$

where we used $|\varphi| \log |\varphi| = o(\lambda)$ in the last inequality. The "moreover" part follows from the fact that $\mathrm{Rec}(\varphi, T, \text{-})$ is a linear function of $s_T$ and that $s_T$ is a linear function of the input $x$.

Now, we prove the soundness of the reduction $R$. Assume that there exists no assignment of weight $2\theta$ that satisfies $\varphi$. We first clarify the condition that the reduction $R$ is successful.

**Claim 7.3.** *With probability at least $1 - o(1)$ over a random choice of $f_1, \dots, f_n$, it holds that*

$$\mathrm{K}(f_B \mid M) \ge |B| \cdot \lambda - 2n \tag{6}$$

*for every $B \subseteq [n]$.*

*Proof.* Fix any $B \subseteq [n]$. By Fact 4.2, we have $\mathrm{K}(f_B \mid M) \ge |B| \cdot \lambda - 2n$ with probability at least $1 - 2^{-2n}$. By a union bound, with probability at least $1 - 2^{-n}$, we obtain $\mathrm{K}(f_B \mid M) \ge |B| \cdot \lambda - 2n$ for every $B \subseteq [n]$. ◇

In what follows, we assume Eq. (6) and prove the soundness of $R$. Let $M$ be a program of size $\theta\lambda$. Let $D$ be a Boolean function that takes an advice string $\alpha = (b, s_1, \dots, s_n)$ and

$((Z_1, Y_1), \ldots, (Z_n, Y_n))$ as input and outputs 1 if and only if $M((Z_1, Y_1 \oplus s_1), \ldots, (Z_n, Y_n \oplus s_n)) = b$. Applying Lemma 6.1 for $\epsilon := 2^{-\lambda^{1-\delta/2}}$, there exists a set $B \subseteq [n]$ such that

$$\mathrm{K}^D(f_B) \leq |B| \cdot (nk + a + O(\log(nka\lambda/\epsilon))), \tag{7}$$

where $a := |\alpha| = O(nk)$, and for every $\alpha \in \{0,1\}^a$,

$$\left|\Pr[D(\alpha, (Z_1, Y_1), \ldots, (Z_n, Y_n)) = 1] - \Pr\big[D(\alpha, (Z_1, Y_1'), \ldots, (Z_n, Y_n')) = 1\big]\right| \leq \epsilon,$$

where $Y_i = f_i \cdot Z_i$ and $Y_i' := Y_i$ if $i \in B$ and $Y_i' \sim \{0,1\}^k$ otherwise. Observe that

$$\mathrm{K}^D(f_B) + O(\log n) \geq \mathrm{K}(f_B \mid M) + O(1) \geq \mathrm{K}(f_B) - |M| \geq |B| \cdot \lambda - |M| - 2n,$$

where the first inequality holds because $D$ can be computed if the program $M$ is given[19] and the last inequality is due to Eq. (6). Combining this inequality with Eq. (7), we obtain

$$|B| \cdot \lambda \cdot (1 - o(1)) \leq |M|,$$

where we used that $nk + a + O(\log(nka\lambda/\epsilon)) = o(\lambda)$. Since $|M| \leq \theta\lambda$, we obtain $|B| \leq (1+o(1)) \cdot \theta < 2\theta \leq \min_{T \in \mathcal{A}_\varphi} |T|$; thus, we obtain that $B \notin \mathcal{A}_\varphi$, i.e., $B$ is not authorized. This implies that

$$\Pr\big[D(\alpha, (Z_1, Y_1'), \ldots, (Z_n, Y_n')) = 1\big] = \Pr\big[M((Z_1, Y_1' \oplus s_1), \ldots, (Z_n, Y_n' \oplus s_n)) = b\big] = \frac{1}{2}$$

by the privacy of the secret sharing scheme. Specifically, $Y'_{[n]\setminus B}$ is uniformly random bits; thus, the input to $M$ depends on only $s_B$, which is statistically independent of a secret $b \sim \{0,1\}$. We conclude that

$$\begin{aligned} \Pr_{(x,b)\sim\mathcal{E}}[M(x) = b] &= \Pr[M((Z_1, Y_1 \oplus s_1), \ldots, (Z_n, Y_n \oplus s_n)) = b] \\ &\leq \frac{1}{2} + \epsilon \leq \frac{1}{2} + 2^{-|x|^{1-\delta}}. \end{aligned}$$

□

## 8 NP-Hardness of Partial MCSP

In order to prove NP-hardness of MCSP*, we need two additional ingredients. First, using the derandomized XOR lemma of Impagliazzo and Wigderson [IW97], we construct a locally-encodable list-decodable error-correcting code.

**Lemma 8.1.** *For any constant $\gamma > 0$, there exist a constant $c \in \mathbb{N}$ and a procedure* Amp *that takes a function $f\colon \{0,1\}^n \to \{0,1\}$ and parameters $\epsilon, \delta \in (0, 1/2), \ell \geq cn$ as input, and returns a function $\mathsf{Amp}^f = \mathsf{Amp}^f_{\epsilon,\delta,\ell}\colon \{0,1\}^\ell \to \{0,1\}$ that satisfies the following properties:*

**List-decodability:** $\mathrm{K}(f) \leq \mathrm{K}_{\frac{1}{2}-\epsilon}(\mathsf{Amp}^f) + 2^{\gamma n} \cdot \mathsf{poly}(1/\epsilon\delta) + 2^n \cdot \mathrm{H}_2(\delta) + O(\ell)$, *where $\mathrm{H}_2$ is the binary entropy function.*

[19]For simplicity, we assume that $M$ is total, i.e., halts in finite steps on every input. This assumption can be removed by considering Kolmogorov complexity relative to the Halting problem $\mathrm{K}^{\mathrm{HALT}}(\text{-})$ instead of $\mathrm{K}(\text{-})$.

**Local encodability:** *There is a nonadaptive $f$-oracle circuit of size* $\mathsf{poly}(n/\epsilon\delta)$ *and depth* $O(\log(n/\epsilon\delta))$ *that computes* $\mathsf{Amp}^f$ *by making* $O(1/\epsilon\delta)$ *queries to* $f$.

**Efficient computability:** $\mathsf{Amp}^f$ *can be computed in time* $\mathsf{poly}(2^n, n/\epsilon\delta)$ *given the truth table of* $f$ *and the parameters as input.*

*Proof.* In order to make sure that $\mathsf{Amp}^f$ can be computed by an $f$-oracle $\mathsf{NC}^1$ circuit, we instantiate the construction of the hardness amplification procedure of [IW97] with a pairwise-independent hitter. Let $S_1, \ldots, S_k \subseteq [d]$ be the design of Proposition 6.2, where $|S_i| = n$ for each $i \in [k]$, $\rho := \gamma n$ (which bounds the size of the intersection of two distinct subsets $S_i, S_j$), and $d = O(\exp(n/\rho) \cdot n^2/\rho) = O(n)$.[20] Let $c \in \mathbb{N}$ be the constant such that $d = (c-3) \cdot n$. Let $\widehat{f}\colon \{0,1\}^{cn} \to \{0,1\}$ be the function such that

$$\widehat{f}(z,r) := f(z_{S_1} \oplus H(r)_1) \oplus \cdots \oplus f(z_{S_k} \oplus H(r)_k),$$

where $z \in \{0,1\}^d$, $r \in \{0,1\}^{3n}$, and $H\colon \{0,1\}^{3n} \to (\{0,1\}^d)^k$ is the pairwise-independent hitter constructed by using Toeplitz matrices [Gol11]. The hitter satisfies the following property: For $k := 1/\epsilon\delta$, for every $X \subseteq \{0,1\}^d$ such that $|X| \geq 2^d \cdot \delta$, with probability at least $1-\epsilon$ over $r \sim \{0,1\}^{3n}$, there exists $i \in [k]$ such that $H(r)_i \in X$.

Now, we define $\mathsf{Amp}^f\colon \{0,1\}^\ell \to \{0,1\}$ by "extending" the function $\widehat{f}$. Specifically, we define $\mathsf{Amp}^f(z,r,x) := \widehat{f}(z,r)$ for $z \in \{0,1\}^d, r \in \{0,1\}^{3n}$, and $x \in \{0,1\}^{\ell - cn}$; in other words, the last $\ell - cn$ bits of the input are ignored in $\mathsf{Amp}^f$. Note that for any $\epsilon' > 0$,

$$\mathrm{K}_{\epsilon'}(\widehat{f}) \leq \mathrm{K}_{\epsilon'}(\mathsf{Amp}^f) + O(\ell) \tag{8}$$

because for a function $g\colon \{0,1\}^\ell \to \{0,1\}$ that agrees with $\mathsf{Amp}^f$ on a $(1-\epsilon')$-fraction of inputs, there exists $x \in \{0,1\}^{\ell - cn}$ such that $g(\text{-}, x)$ agrees with $\widehat{f}$ on a $(1-\epsilon')$-fraction of inputs. It is easy to see the efficient computability of $\mathsf{Amp}^f$.

To see the local encodability, observe that for each $i \in [k]$, each bit of $H(r)_i \in \{0,1\}^d$ can be computed in $\mathsf{NC}^1$ on input $r \in \{0,1\}^{3n}$. Thus, there exists a nonadaptive $f$-oracle circuit of size $\mathsf{poly}(nk)$ and depth $O(\log nk)$ that computes $\mathsf{Amp}^f$.

To see the list-decodability, we use the property of the hardness amplification procedure of Impagliazzo and Wigderson [IW97]. They showed that for every function $g\colon \{0,1\}^n \to \{0,1\}$ that agrees with $\widehat{f}$ on a $(1/2+\epsilon)$-fraction of inputs, there exists a $g$-oracle program of size $2^{\gamma n} \cdot \mathsf{poly}(1/\epsilon\delta)$ that agrees with $f$ on a $(1-\delta)$-fraction of inputs (see also [HVV06; Hir20a] for an exposition). Thus, given a program that describes at least a $(1/2+\epsilon)$-fraction of $\widehat{f}$, we can construct a program that describes at least a $(1-\delta)$-fraction of $f$ and obtain that

$$\mathrm{K}_\delta(f) \leq \mathrm{K}_{\frac{1}{2}-\epsilon}(\widehat{f}) + 2^{\gamma n} \cdot \mathsf{poly}(1/\epsilon\delta). \tag{9}$$

Let $h$ be the program of size $\mathrm{K}_\delta(f)$ that agrees with $f$ on a $(1-\delta)$-fraction of inputs. Since the Hamming weight of the truth table of $f \oplus h$ is at most $\delta 2^n$, we have

$$\mathrm{K}(f \oplus h) \leq \log\left(\sum_{i=0}^{\delta 2^n} \binom{2^n}{i}\right) + O(\log n) \leq 2^n \cdot \mathrm{H}_2(\delta) + O(\log n).$$

[20] We may assume without loss of generality that $k := 1/\epsilon\delta \leq 2^n$, as otherwise the upper bound in the list-decodability is trivial.

Since $f$ can be described by $h$ and $f \oplus h$, we obtain that

$$\mathrm{K}(f) \le \mathrm{K}_\delta(f) + 2^n \cdot \mathrm{H}_2(\delta) + O(n). \tag{10}$$

The property of the list-decodability follows from Eqs. (8) to (10).

$\square$

Second, we use Uhlig's theorem to ensure that $\mathsf{Amp}^f$ can be computed by a circuit of size $O(2^n/n)$ for any function $f\colon \{0,1\}^n \to \{0,1\}$.

**Lemma 8.2** (Uhlig [Uhl74; Uhl92]; see also [Weg87])**.** *Let $r\colon \mathbb{N} \to \mathbb{N}$ be a function such that $r(n) = 2^{o(n/\log n)}$. Then, for all large $n \in \mathbb{N}$, for any function $f\colon \{0,1\}^n \to \{0,1\}$, there exists a circuit of size $(1+o(1))\cdot 2^n/n$ and of depth $(1+o(1))\cdot n$ that computes $f^{r(n)}\colon (\{0,1\}^n)^{r(n)} \to \{0,1\}^{r(n)}$, i.e., the $r(n)$-wise direct product of $f$.*

We now present a reduction that will be used to prove $\mathsf{NP}$-hardness of MCSP$^*$.

**Lemma 8.3.** *There exists a randomized polynomial-time reduction $R$ that takes a* CMMSA *instance $(\Phi, w, \theta)$ of size $n$ and degree $\Delta$ and a parameter $\epsilon_0 > 0$ such that $\epsilon_0/\Delta \le n^{o(1/\log\log n)}$ as input and outputs a distribution $\mathcal{E}$ (represented by a circuit) and $s \in \mathbb{N}$ with the following properties:*

**Completeness:** *If there exists an assignment of weight $\theta$ that satisfies all the formulas in $\Phi$, then there exist an $O(s)$-time program $M$ of size $O(s)$ and a circuit $C$ of size $O(\frac{s}{\log s})$ and depth $O(\log s)$ such that*

$$\Pr_{(x,b)\sim\mathcal{E}}[M(x) = b] = 1 \quad \textit{and} \quad \Pr_{(x,b)\sim\mathcal{E}}[C(x) = b] = 1.$$

**Soundness:** *If every assignment of weight $\theta$ satisfies at most an $\epsilon_0$-fraction of the formulas in $\Phi$, then with probability $1 - o(1)$ over the internal randomness of $R$, for every program of size $s/2$,*

$$\Pr_{(x,b)\sim\mathcal{E}}[M(x) = b] \le \frac{1}{2} + 2\epsilon_0.$$

*Moreover, we have $|x| = O(\log n + \Delta^2)$, and all the elements in $\mathrm{supp}(\mathcal{E})$ can be enumerated in time $2^{O(|x|)}$.*

*Proof.* Let $(\Phi, w, \theta)$ be an instance of CMMSA, where $\Phi = \{\varphi_1, \dots, \varphi_\nu\}$ is a degree-$\Delta$ collection of monotone formulas over the set $[n]$ of input variables and $w\colon [n] \to \mathbb{N}$ is a weight function. For each $j \in [\nu]$, let $V_j$ denote the set $\left\{v_1^j < \cdots < v_m^j\right\} \subseteq [n]$ of the variables of $\varphi_j$. Here, $m \le \Delta$ is the number of variables on which $\varphi_j$ depends for every $j \in [\nu]$.[21]

Let $\mathsf{Amp}$ be the hardness amplification procedure of Lemma 8.1 for $\gamma := 1/2$, $\delta := 1/\log n$, $\epsilon := \epsilon_0/2\Delta^2$, and let $c \ge 1$ be the constant of Lemma 8.1. Let $\lambda \in \mathbb{N}$ be a sufficiently large parameter; specifically, we may define

$$\lambda := \max\left\{m\Delta, (n\mu)^2, \theta, (n\Delta w_{\max})^4\right\},$$

[21]We may assume without loss of generality that the number of variables in $\varphi_j$ is the same for all $j \in [\nu]$.

where $w_{\max} := \max\{w(k) \mid k \in [n]\} \geq 1$. We also define

$$\ell := c \cdot \lceil \log(w_{\max}\lambda) \rceil.$$

The reduction $R$ operates as follows. For each $k \in [n]$, the reduction $R$ picks $f_k \sim \{0,1\}^{w(k)\cdot\lambda}$ randomly. We identify the string $f_k \in \{0,1\}^{w(k)\cdot\lambda}$ with a function $f_k \colon \{0,1\}^{\lceil \log |f_k| \rceil} \to \{0,1\}$; let $\widehat{f_k} := \mathsf{Amp}^{f_k} \colon \{0,1\}^\ell \to \{0,1\}$ denote the hardness-amplified version of the function $f_k$. Then, it constructs a distribution $\mathcal{E} = \mathcal{E}(f_1, \dots, f_n)$ from which one can sample $(x, b)$ as follows: Choose a formula $\varphi_j \sim \Phi$ randomly. Choose a secret $b \sim \{0,1\}$ randomly. Using Lemma 4.5, we share the secret $b$ among $m$ parties; that is, let $(s_1, \dots, s_m) := \mathrm{Share}(\varphi_j, b)$. We may assume without loss of generality that the length of each share $s_i$ is $\Delta$ because $|\varphi_j| \leq \Delta$. Let $\mathcal{S} = (S_1, \dots, S_{m\Delta})$ be the collection of $\ell$-sized subsets of $[d]$ from Proposition 6.2, where $d = O(\ell)$ and $\rho := \ell/2c$. Here, we used that $m\Delta \leq \lambda \leq 2^\ell$. Let $\mathcal{S}_i := (S_{(i-1)\Delta+1}, \dots, S_{(i-1)\Delta+\Delta})$ for every $i \in [m]$. Define a string $x \in \{0,1\}^{O(\log \nu)} \times \{0,1\}^d \times (\{0,1\}^\Delta)^m$ to be

$$\left(j, z, \mathrm{NW}_{\mathcal{S}_1}\left(\widehat{f_{v_1^j}}; z\right) \oplus s_1, \dots, \mathrm{NW}_{\mathcal{S}_m}\left(\widehat{f_{v_m^j}}; z\right) \oplus s_m\right).$$

Here, "$u \oplus v$" denotes the bit-wise XOR. The output of the distribution $\mathcal{E}$ is defined to be $(x, b)$. The output of the reduction $R$ is defined to be the circuit that represents $\mathcal{E}$ and the parameter $s := \theta \cdot \lambda$. Observe that $|x| = O(\log \nu + d + \Delta m) = O(\log n + \Delta^2)$. In what follows, we prove the correctness of the reduction $R$.

To see the completeness of the reduction $R$, assume that there exists an assignment $\alpha \colon [n] \to \{0,1\}$ such that $\Pr_{\varphi \sim \Phi}[\varphi(\alpha) = 1] = 1$ and $w(\alpha) \leq \theta$. Let $T = \{i \in [n] \mid \alpha(i) = 1\}$. We construct a small program $M$ that correctly computes $b$ given $x$ as input for every $(x, b) \in \mathrm{supp}(\mathcal{E})$. We hard-wire the sets $T$, $\{f_k \mid k \in T\}$ and $\{(j, V_j) \mid j \in [\nu]\}$ into $M$. The program $M$ takes $x = (j, z, \xi_1, \dots, \xi_m)$ as input, lets $s_i := \mathrm{NW}_{\mathcal{S}_i}\left(\widehat{f_{v_i^j}}; z\right) \oplus \xi_i$ for each $v_i^j \in V_j \cap T$, and reconstructs and outputs the secret $b := \mathrm{Rec}(\varphi_j, V_j \cap T, s_{V_j \cap T})$. Since $\varphi_j(\chi_{V_j \cap T}) = \varphi_j(\chi_T) = \varphi_j(\alpha) = 1$, the set $V_j \cap T$ is authorized with respect to the access structure represented by $\varphi_j$. The correctness of the secret sharing scheme $(\mathrm{Share}(\varphi_j, \text{-}), \mathrm{Rec}(\varphi_j, \text{-}))$ implies the correctness of $M$, i.e.,

$$\Pr_{(x,b) \sim \mathcal{E}}[M(x) = b] = 1.$$

The bottleneck of the running time of $M$ is the computation of $\widehat{f_k}$ given oracle access to $f_k$, which takes at most $\mathsf{poly}(\ell/\epsilon\delta) \leq \lambda^{o(1)} \ll s$ time by Lemma 8.1. (Here, we measure the running time of $M$ in the sense of KT-complexity; i.e, we assume that $M$ is given oracle access to the hard-wired inputs.) Overall, $M$ runs in time $O(s)$. Moreover, the size $|M|$ of the program $M$ is at most

$$|T| \cdot O(\log n) + \sum_{k \in T} |f_k| + \nu n \leq w(T) \cdot \lambda + o(\lambda) \leq (1 + o(1)) \cdot s.$$

We now construct a circuit $C$. The construction is similar to $M$, except that we need to carefully hard-wire a circuit that computes the $\Delta$-wise direct product of $\widehat{f_k}$ in $C$. Let $x = (j, z, \xi_1, \dots, \xi_m)$ be an input to $C$. First, for each $k \in T$, the circuit $C$ computes strings $(y_1^k, \dots, y_\Delta^k) \in \left(\{0,1\}^\ell\right)^\Delta$ such that if there exists $i \in [m]$ such that $k = v_i^j$, then $y_p^k = z_{S_{(i-1)\Delta+p}}$ for every $p \in [\Delta]$. Then, for each $k \in T$, the circuit $C$ computes $(\widehat{f_k}(y_1^k), \dots, \widehat{f_k}(y_\Delta^k))$ from $(y_1^k, \dots, y_\Delta^k)$. By Lemma 8.1, each

$\widehat{f}_k(y_p^k)$ can be computed by $O(1/\epsilon\delta)$ nonadaptive queries to $f$; thus, the tuple $\left(\widehat{f}_k(y_p^k) \;\middle|\; p \in [\Delta]\right)$ can be computed by $O(\Delta/\epsilon\delta)$ nonadaptive queries to $f_k$. By Lemma 8.2, the $O(\Delta/\epsilon\delta)$ nonadaptive queries to $f_k$ can be simulated by a circuit of size $O(2^{\log|f_k|}/\log|f_k|)$, as long as $\Delta/\epsilon\delta \le \lambda^{o(1/\log\log\lambda)}$, which is satisfied under our assumption. Overall, the tuple $\left(\widehat{f}_k(y_p^k) \;\middle|\; p \in [\Delta]\right)$ can be computed by a circuit of size

$$O(2^{\log|f_k|}/\log|f_k|) + \mathsf{poly}(\Delta\log|f_k|/\epsilon\delta) \le O(w(k)\cdot\lambda/\log\lambda)$$

Here, we used that $\mathsf{poly}(\Delta\log(w_{\max}\lambda)/\epsilon\delta) \le \lambda^{o(1)}$ and $w(k) \ge 1$. Finally, $C$ computes $s_i := (\widehat{f}_k(y_1^k), \ldots, \widehat{f}_k(y_\Delta^k)) \oplus \xi_i$ for every $k$ and $i$ such that $k = v_i^j$. Then, $C$ outputs $b = \mathrm{Rec}(\varphi_j, V_j \cap T, s_{V_j\cap T})$. The bottleneck of the circuit size of $C$ is the computation of the $\Delta$-wise direct product of $\widehat{f}_k$; overall, the circuit size is at most

$$\sum_{k\in T} O(w(k)\cdot\lambda/\log\lambda) \le O(w(T)\lambda/\log\lambda) \le O(\theta\lambda/\log\lambda) \le O(s/\log s),$$

where we used that $s = \theta\lambda \le \lambda^2$ in the last inequality. The depth of $C$ is at most $O(\log s)$.

To see the soundness of the reduction $R$, we clarify the condition that the reduction $R$ succeeds. The condition is that $\mathrm{K}(f_{[n]}) \ge \lambda\cdot w([n]) - \log n$, which happens with probability at least $1-\frac{1}{n}$ by Fact 4.2. Note that under this condition, we also have

$$\lambda\cdot w(T) \le \mathrm{K}(f_T) + O(|T|\cdot\log n) \tag{11}$$

for every $T \subseteq [n]$ because $\mathrm{K}(f_{[n]}) \le \mathrm{K}(f_T) + \lambda\cdot w([n]\setminus T) + O(|T|\cdot\log n)$. In what follows, we assume Eq. (11) and prove the soundness of the reduction $R$.

Consider an arbitrary program $M$ of size at most $\frac{s}{2}$. We claim that $M$ fails to compute $b$ on input $x$ with probability $\approx \frac{1}{2}$ over a random choice of $(x,b) \sim \mathcal{E}$. We define $D$ to be a function that checks whether $M$ succeeds to compute $b$ on input $x$; specifically, the function $D$ takes an advice string $\alpha = (b, j, s_1, \ldots, s_m, V_j)$, a seed $Z \in \{0,1\}^d$, and $(Y_1,\ldots,Y_n) \in (\{0,1\}^\Delta)^n$, and outputs 1 if and only if $M(j, Z, Y_{v_1^j}\oplus s_1, \ldots, Y_{v_m^j}\oplus s_m) = b$. Applying Lemma 6.6 to $\Delta n$ strings $(\widehat{f}_1,\ldots,\widehat{f}_1),\ldots,(\widehat{f}_n,\ldots,\widehat{f}_n)$, where each $\widehat{f}_k$ is repeated $\Delta$ times, we obtain a subset $B \subseteq [n]$ such that for any $\alpha$,

$$\left|\Pr[D(\alpha, Z, Y_1,\ldots,Y_n)=1] - \Pr\big[D(\alpha, Z, Y_1',\ldots,Y_n')=1\big]\right| \le \epsilon_0, \tag{12}$$

where $Z \sim \{0,1\}^d$, $Y_k$ is identical to $\mathrm{NW}_{\mathcal{S}_k}(\widehat{f}_k; Z)$ and $Y_k'$ is identical to $Y_k$ if $k \in B$ and to the uniform distribution over $\{0,1\}^\Delta$ if $k \in [n]\setminus B$. Moreover, since $D(\alpha, \text{-})$ depends on at most $m\Delta$ of the bits of $Y_1,\ldots,Y_n$, for every $k \in B$, it holds that

$$\mathrm{K}^D_{\frac{1}{2}-\frac{\epsilon_0}{2m\Delta}}(\widehat{f}_k) \le n\Delta\cdot 2^\rho + |\alpha| + O(\log(2^\ell\Delta n|\alpha|d/\epsilon_0)) \le n\Delta\cdot 2^{\lceil\log(w_{\max}\cdot\lambda)\rceil/2} + o(\lambda) \le o(\lambda).$$

Here, we used that the length of the advice string $\alpha$ is at most $O(\log\nu) + d + m\Delta + n \le o(\lambda)$ in the second inequality, and that $(n\Delta w_{\max})^4 \le \lambda$ in the last inequality. By Lemma 8.1, we also have

$$\mathrm{K}^D(f_k) \le \mathrm{K}^D_{\frac{1}{2}-\epsilon}(\widehat{f}_k) + |f_k|^{1/2}\cdot\mathsf{poly}(1/\epsilon\delta) + \mathrm{H}_2(\delta)\cdot|f_k| + O(\ell) \le \mathrm{K}^D_{\frac{1}{2}-\frac{\epsilon_0}{2m\Delta}}(\widehat{f}_k) + o(w(k)\cdot\lambda)$$

because $\mathsf{poly}(1/\epsilon\delta) \le \lambda^{o(1)}$ and $\mathrm{H}_2(\delta) \le o(1)$. Combining the two inequalities above, we obtain

$$\mathrm{K}^D(f_k) \le o(\lambda\cdot w(k))$$

for every $k \in B$. By summing this inequality over all $k \in B$, we get

$$\mathrm{K}^D(f_B) \le o(\lambda \cdot w(B)).$$

Since the function $D$ can be computed using $M$, we have

$$\mathrm{K}(f_B) - |M| \le \mathrm{K}(f_B \mid M) + O(1) \le \mathrm{K}^D(f_B) + O(1) \le o(\lambda \cdot w(B)).$$

Using Eq. (11), we conclude that

$$(1 - o(1)) \cdot \lambda \cdot w(B) \le |M| \le \frac{s}{2} = \frac{\theta\lambda}{2},$$

which implies $w(B) < \theta$. It follows from the assumption that $\chi_B$ satisfies at most an $\epsilon_0$-fraction of the formulas in $\mathit{\Phi}$; that is,

$$\Pr_{\varphi\sim\mathit{\Phi}}[\varphi(\chi_B) = 1] \le \epsilon_0.$$

By Eq. (12) and the definition of $D$, we have

$$\left|\Pr\left[M(j, Z, Y_{v_1^j} \oplus s_1, \dots, Y_{v_m^j} \oplus s_m) = b\right] - \Pr\left[M(j, Z, Y'_{v_1^j} \oplus s_1, \dots, Y'_{v_m^j} \oplus s_m) = b\right]\right| \le \epsilon_0$$

for every $j \in [\nu]$, every $(s_1, \dots, s_m) \in \left(\{0,1\}^{\Delta}\right)^m$, and every $b \in \{0,1\}$. The distribution of the first term is equivalent to the distribution of the input $x$ if we take an average over $j \sim [\nu]$, $b \sim \{0,1\}$, and $(s_1, \dots, s_m) = \mathrm{Share}(\varphi_j, b)$. In the second term, the share $s_i$ can be "removed" for every $i \in [m]$ such that $v_i^j \in [n] \setminus B$, as $Y'_{v_i^j} \oplus s_i$ is statistically identical to the uniform distribution; thus, $M$ takes as input only the shares $s_{B\cap V_j}$, which is statistically independent of the secret $b \sim \{0,1\}$ if $\varphi_j(\chi_{B\cap V_j}) = 0$ because of the privacy of the secret sharing scheme. Since $\varphi_j(\chi_B) = \varphi_j(\chi_{B\cap V_j})$, if $\varphi_j(\chi_B) = 0$, we obtain

$$\Pr\left[M(j, Z, Y'_{v_1^j} \oplus s_1, \dots, Y'_{v_m^j} \oplus s_m) = b\right] = \frac{1}{2}.$$

We conclude that

$$\begin{aligned}
&\Pr_{(x,b)\sim\mathcal{E}}[M(x) = b] \\
\le\ &\Pr_{\varphi\sim\mathit{\Phi}}[\varphi(\chi_B) = 1] + \Pr\left[M(j, Z, Y_{v_1^j} \oplus s_1, \dots, Y_{v_m^j} \oplus s_m) = b \;\middle|\; \varphi_j(\chi_B) = 0\right] \\
\le\ &\epsilon_0 + \frac{1}{2} + \epsilon_0,
\end{aligned}$$

which completes the proof of the soundness. □

We give a formal definition of meta-computational problems. We regard a function $f\colon \{0,1\}^n \to \{0,1,*\}$ as a partial function such that $f$ is undefined on any input $x \in \{0,1\}^n$ such that $f(x) = *$. The *truth table* of a (partial) function $f$ is the string (over $\{0,1,*\}$) obtained by concatenating $f(x)$ over all inputs $x \in \{0,1\}^n$ in the lexicographical order. We often identify a function with its truth table. We say that a function $g\colon \{0,1\}^n \to \{0,1\}$ is *consistent* with a partial function $f\colon \{0,1\}^n \to \{0,1,*\}$ if $g(x) = f(x)$ for every $x \in f^{-1}(\{0,1\})$. A string over $\{0,1,*\}$ is said to be *partial.* The notion of the consistency between a function and a partial function naturally extends to the notion of the consistency between a string and a partial string.

**Definition 8.4** (Partial variants of meta-computational problems)**.**

- *Let* MCSP$^*$ *denote the language consisting of pairs* $(f, 1^s)$ *such that there exists a circuit of size* $s$ *that computes some function consistent with the partial function* $f \colon \{0,1\}^n \to \{0,1,*\}$*.*
- *Let* NC$^1$-MCSP$^*$ *denote the language identical to* MCSP$^*$*, except that the depth of size-*$s$ *circuits is restricted to be* $c \cdot \log s$*, where* $c$ *is a sufficiently large universal constant.*[22]
- *Let* MKTP$^*$ *denote the language consisting of pairs* $(x, 1^s)$ *such that* $x \in \{0,1,*\}^n$ *for some* $n \in \mathbb{N}$ *and there exists a string* $y \in \{0,1\}^n$ *consistent with* $x$ *and* $\mathrm{KT}(y) \le s$*.*
- *Let* MINKT$^*$ *denote the language consisting of pairs* $(x, 1^t, 1^s)$ *such that* $x \in \{0,1,*\}^n$ *for some* $n \in \mathbb{N}$ *and there exists a string* $y \in \{0,1\}^n$ *consistent with* $x$ *and* $\mathrm{K}^t(y) \le s$*.*

**Theorem 8.5.** MCSP$^*$*,* NC$^1$-MCSP$^*$*,* MKTP$^*$*, and* MINKT$^*$ *are* NP*-hard under randomized polynomial-time many-one reductions. Moreover, it is* NP*-hard to approximate these problems within a factor of* $(\log N)^\alpha$ *on inputs of length* $N$ *for some constant* $\alpha > 0$*.*

*Proof.* Let $L$ be an arbitrary language in NP. Combining NP-hardness of CMMSA (Theorem 5.2) for the parameter $\Delta(n) := (\log n)^{1/2}$ with the reduction of Lemma 8.3, we obtain a reduction $R$ from $L \in \mathsf{NP}$. On inputs of length $n$, the combined reduction $R$ outputs a distribution $\mathcal{E}$ and a parameter $s \in \mathbb{N}$ such that $\mathrm{supp}(\mathcal{E})$ can be enumerated in polynomial time and $\mathrm{supp}(\mathcal{E}) \subseteq \{0,1\}^{\log N} \times \{0,1\}$ for some $N = n^{O(1)}$. Using $\mathcal{E}$, we construct a partial Boolean function $f \colon \{0,1\}^{\log N} \to \{0,1,*\}$ as follows: $f(x) := b$ if there exists $(x,b) \in \mathrm{supp}(\mathcal{E})$ and $f(x) := *$ otherwise.[23] The output of the reduction to a meta-computational problem $P \in \{\mathrm{MCSP}^*, \mathrm{NC}^1\text{-}\mathrm{MCSP}^*, \mathrm{MKTP}^*\}$ is defined to be $(f, 1^{s_P})$, where we identify $f$ with a string $f \in \{0,1,*\}^N$. (The proof for $P = \mathrm{MINKT}^*$ is similar and thus omitted.) Here, we define $s_{\mathrm{MCSP}^*} = s_{\mathrm{NC}^1\text{-}\mathrm{MCSP}^*} := O(s/\log s)$ and $s_{\mathrm{MKTP}^*} := O(s)$. Let $R'$ denote the randomized polynomial-time reduction that takes an instance of $L$ and produces the instance $(f, 1^{s_P})$.

In what follows, we prove the correctness of the reduction $R'$. To prove the completeness of the reduction, let $\varphi$ be a Yes instance of $L$ and let $(f, 1^{s_P})$ be the instance produced by the reduction $R'$ on input $\varphi$. By the completeness property of Lemma 8.3, there exists a circuit $C$ of size $O(s/\log s) \le s_{\mathrm{NC}^1\text{-}\mathrm{MCSP}^*}$ and depth $O(\log s) \le c \cdot \log s_{\mathrm{NC}^1\text{-}\mathrm{MCSP}^*}$, where $c$ is a universal constant, such that the function computed by $C$ is consistent with $f$. Thus, $(f, 1^{s_{\mathrm{NC}^1\text{-}\mathrm{MCSP}^*}})$ is a Yes instance of MCSP$^*$ and NC$^1$-MCSP$^*$. Similarly, there exists a program $M$ of size $O(s) \le s_{\mathrm{MKTP}^*}$ such that the function computed by $M$ is consistent with $f$, which implies that $(f, 1^{s_{\mathrm{MKTP}^*}})$ is a Yes instance of MKTP$^*$.

We now prove the soundness of the reduction $R'$. Assume that $\varphi \notin L$ and $R$ outputs a distribution $\mathcal{E}$ on input $\varphi$. By the soundness property of Lemma 8.3 and the gap of CMMSA, for every program $M$ of size $s \cdot g$, where $g = \Delta(n)^\alpha = (\log n)^{\alpha/2}$ for some constant $\alpha > 0$, we have

$$\Pr_{(x,b) \sim \mathcal{E}}[M(x) = b] \le \frac{1}{2} + 2(\log n)^{-\alpha/2}.$$

In particular, for every $y \in \{0,1\}^N$ consistent with $f$, we obtain $\mathrm{K}(y) \ge s \cdot g$. This completes the proof of NP-hardness of MKTP$^*$ and MINKT$^*$.

[22]We prove NP-hardness of NC$^1$-MCSP$^*$ for a sufficiently large universal constant $c$.

[23]If there exist two conflicting samples $(x,0), (x,1) \in \mathrm{supp}(\mathcal{E})$, then it is impossible to satisfy the perfect completeness; thus, such an instance can be immediately rejected.

It remains to argue the case of MCSP$^*$ and NC$^1$-MCSP$^*$. Assume that there exists a circuit $C$ of size $s'$ such that the function computed by $C$ is consistent with $f$. Let $y$ be the truth table of $C$. Since $C$ can be described by using $O(s' \log s')$ bits, we obtain

$$s \cdot g \le \mathrm{K}(y) \le O(s' \log s'),$$

which implies that $s' \ge \Omega(sg/\log s') \ge \Omega(g \cdot s/\log s) \ge \Omega(g) \cdot s_{\mathrm{NC}^1\text{-MCSP}^*}$. This means that NC$^1$-MCSP$^*$ and MCSP$^*$ are NP-hard to approximate within a factor of $\Omega(g) = (\log n)^{\Omega(1)}$. ☐

Theorem 1.2 can be proved in the same way with Theorem 8.5.

Finally, we prove NP-hardness of AveMCSP, i.e., the average-case variant of MCSP with respect to the uniform distribution. The proof of Theorem 8.5 shows that another average-case varinat of MCSP$^*$, which takes a distribution $\mathcal{E}$ as input and asks the average-case circuit complexity of $f$ with respect to the distribution $\mathcal{E}$, is also NP-hard. We show that the distribution can be made uniform.

**Definition 8.6** ([San20])**.** AveMCSP *is the language consisting of the truth table of a function* $f\colon \{0,1\}^n \to \{0,1\}$*, parameters* $s \in \mathbb{N}$ *and* $\delta \in \{i \cdot 2^{-n} \mid i \in \mathbb{N}\}$ *such that there exists a circuit* $C$ *of size* $s$ *such that*

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

**Theorem 8.7.** AveMCSP *is* NP*-hard under randomized polynomial-time many-one reductions. Similarly, it is* NP*-hard (under randomized reductions) to compute the average-case time-bounded Kolmogorov complexity* $\mathrm{K}^t_\delta(x)$ *on input* $(x, t, \delta)$*.*

*Proof.* The idea is to replace $f(x) = *$ with a uniformly random bit $f(x) \sim \{0,1\}$ in the proof of NP-hardness of MCSP$^*$.

Let $R$ be the reduction of Theorem 8.5 that shows NP-hardness of MCSP$^*$. Given an instance of CMMSA, the reduction $R$ outputs a partial function $f\colon \{0,1\}^n \to \{0,1,*\}$ and a size parameter $s$. Let $D := f^{-1}(\{0,1\})$ be the domain of $f$ and let $\overline{D} := \{0,1\}^n \setminus D$. Define $\beta := 2^{-\Delta^2-3}$, where $\Delta$ is the degree of the CMMSA instance. Inspecting the proofs of Lemma 8.3 and Theorem 8.5, one can observe that the following properties are satisfied.

1. $\beta \cdot 2^{n+3} \le |D| \le 2^{n-1}$.
2. $s \log s \le o(\beta^2 \cdot 2^n)$.
3. The distribution $\mathcal{E}$ of Lemma 8.3 is identical to the distribution of $(X, f(X))$, where $X$ is the uniform distribution over $D$.

Let $f'\colon \{0,1\}^n \to \{0,1\}$ be a random function such that $f'(x) := f(x)$ if $x \in D$ and otherwise $f'(x) \sim \{0,1\}$ is defined to be a uniformly random bit. The NP-hardness reduction for AveMCSP outputs the truth table of $f'$ and the parameters $s$ and $\delta := \frac{1}{2} - |D| \cdot 2^{-n-2}$.

We prove the correctness of the reduction. In the Yes case, there exists a circuit $C$ of size $s$ such that

$$\Pr_{x \sim D}[C(x) = f(x)] = 1.$$

For every $x \in \overline{D}$, let $I_x \in \{0,1\}$ denote the random variable that takes 1 if and only if $f'(x) = C(x)$. Note that $\mathbb{E}[I_x] = \frac{1}{2}$. Since

$$\Pr_{x \sim \overline{D}}\bigl[C(x) = f'(x)\bigr] = \frac{1}{2^n - |D|} \sum_{x \in \overline{D}} I_x,$$

by the Chernoff bound, with probability at least $1 - o(1)$ over the random choice of $f'$, we obtain

$$\Pr_{x \sim \overline{D}}\bigl[C(x) = f'(x)\bigr] \geq \frac{1}{2} - \beta.$$

Under this event, we have

$$\begin{aligned}
&\Pr_{x \sim \{0,1\}^n}\bigl[C(x) = f'(x)\bigr] \\
&\geq \Pr_{x \sim \{0,1\}^n}[x \in D] \cdot 1 + \Pr_{x \sim \{0,1\}^n}\bigl[x \in \overline{D}\bigr] \cdot \left(\frac{1}{2} - \beta\right) \\
&\geq \frac{1}{2} + |D| \cdot 2^{-n-1} - \beta \geq 1 - \delta.
\end{aligned}$$

This shows that $(f', s, \delta)$ is a Yes instance of AveMCSP.

Now, consider the No case. In this case, for every circuit of size $s' := s \cdot \Delta^{\alpha}$,

$$\Pr_{x \sim D}[C(x) = f(x)] \leq \frac{1}{2} + \Delta^{-\alpha} < \frac{1}{2} + 2^{-3}.$$

By the Chernoff bound, with probability at least $1 - \exp(-\Omega(\beta^2 2^n))$ over the random choice of $f'$, it holds that

$$\Pr_{x \sim \overline{D}}\bigl[C(x) = f'(x)\bigr] \leq \frac{1}{2} + \beta.$$

By taking a union bound over all the circuits of size $s \leq s'$, with probability at least $1 - \exp(-\Omega(\beta^2 2^n) + O(s \log s)) \geq 1 - o(1)$ over $f'$, it holds that for every circuit $C$ of size $s$,

$$\Pr_{x \sim \overline{D}}\bigl[C(x) = f'(x)\bigr] \leq \frac{1}{2} + \beta.$$

Under this event, we have

$$\begin{aligned}
&\Pr_{x \sim \{0,1\}^n}\bigl[C(x) = f'(x)\bigr] \\
&< \Pr_{x \sim \{0,1\}^n}[x \in D] \cdot \left(\frac{1}{2} + 2^{-3}\right) + \Pr_{x \sim \{0,1\}^n}\bigl[x \in \overline{D}\bigr] \cdot \left(\frac{1}{2} + \beta\right) \\
&\leq \frac{1}{2} + |D| \cdot 2^{-n-3} + \beta \leq 1 - \delta.
\end{aligned}$$

This shows that $(f', s, \delta)$ is a No instance of AveMCSP. $\square$

## A Weak Learning by Small Hypotheses in Heuristica

Hirahara and Nanashima [HN21] showed the feasibility of strong agnostic learning from average-case easiness of NP. Applying their proof techniques to weak learning, we observe that size-$s$ programs on $n$ inputs can be weakly learned by hypotheses of size $s \cdot 1.01n$. Consequently, extending the inapproximability factor $n^\epsilon$ of Theorem 1.1 to $1.01n$ is sufficient to exclude Heuristica. This indicates that our NP-hardness result comes somewhat close to the setting of improper learning.

For some technical reason, we impose a mild restriction on the type of NP-hardness reductions. We consider a reduction $R$ that outputs a distribution $\mathcal{E}$ and a size parameter $s$, as in Theorem 1.1. We say that $R$ is *size-expanding* if $R$ outputs $(\mathcal{E}, 1^s)$ on input $\varphi$ such that $s = \omega(|\varphi|)$. That is, we require that the size parameter is much larger than the length of inputs. All the NP-hardness reductions in this paper are size-expanding. We now state the main result of this section.

**Proposition A.1.** *Assume that for some polynomial $t_1$, for any polynomial $t_2$, there exists a randomized polynomial-time size-expanding reduction from an* NP*-complete problem to the following promise problem: The input consists of a size parameter $s$ represented in unary and a distribution $\mathcal{E}$ represented by a circuit such that* $\mathrm{supp}(\mathcal{E}) \subseteq \{0,1\}^n \times \{0,1\}$ *for some $n \in \mathbb{N}$. The task is to distinguish the following two cases:*

**Yes:** *There exists a $t_1(n)$-time program $M$ of size $s$ such that*

$$\Pr_{(x,b)\sim\mathcal{E}}[M(x) = b] = 1.$$

**No:** *For any $t_2(n)$-time program $M$ of size $s \cdot 1.01n$,*

$$\Pr_{(x,b)\sim\mathcal{E}}[M(x) = b] < \frac{1}{2} + \frac{1}{5s}.$$

*Then, Heuristica does not exist: i.e.,* $\mathsf{P} \neq \mathsf{NP}$ *if and only if* $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$.

Here, $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ means that for every problem in NP and for every polynomial-time samplable distribution over instances, there exists an average-case polynomial-time algorithm. See [BT06a; Hir21] for background on average-case complexity.

Recall the notion of sampling depth.

**Definition A.2** ([HN21])**.** *For parameters $t, m \in \mathbb{N}$, the* sampling depth *of a distribution $\mathcal{E}$ is*

$$\mathrm{sd}^t_m(\mathcal{E}) := \mathop{\mathbb{E}}_{\forall i,\ (x_i,b_i)\sim\mathcal{E}}\big[\mathrm{K}^t(x_1,\ldots,x_m) - \mathrm{K}(x_1,\ldots,x_m)\big].$$

Hirahara and Nanashima [HN21] showed that any distribution with small sampling depth can be efficiently learned. We prove that there exists a weak learning algorithm that produces a hypothesis of size $(1 + o(1)) \cdot sn$, provided that $\mathrm{sd}^t_m(\mathcal{E}) = o(s)$. This assumption will be justified later.

**Lemma A.3.** *If* $\mathsf{DistNP} \subseteq \mathsf{AvgP}$*, then there exist a polynomial-time algorithm $M$ and a polynomial $p$ such that, on input $(\mathcal{E}, 1^t, 1^s)$, where $\mathcal{E}$ is a distribution represented by a circuit, $M$ decides the following two cases:*

**Yes:** *There exists a $t$-time program $M$ of size $s$ such that*

$$\Pr_{(x,b)\sim\mathcal{E}}[M(x)=b]=1.$$

**No:** *For any $p(t,m)$-time program $M$ of size $mn+m+O(\log ts)$,*

$$\Pr_{(x,b)\sim\mathcal{E}}[M(x)=b]<\frac{1}{2}+\frac{1}{5m},$$

*where $m$ is an arbitrary parameter such that $m \geq s + 2\mathrm{sd}^t_m(\mathcal{E}) + O(\log mnt)$.*

*Proof.* Under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$, Hirahara [Hir18; Hir20a] showed that there exist a polynomial-time algorithm $\widetilde{\mathrm{K}}$ and a polynomial $p$ such that

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

for every $x \in \{0,1\}^*$ and every $t \geq |x|$.

We present a randomized polynomial-time algorithm $V$, which can be derandomized using the theorem of Buhrman, Fortnow, and Pavan [BFP05]. On input $(\mathcal{E},1^t,1^s)$, the algorithm $V$ draws $m$ samples $(x_1,b_1),\ldots,(x_m,b_m)$ from $\mathcal{E}$ independently and outputs 1 if and only if

$$\widetilde{\mathrm{K}}(x_1,\ldots,x_m,b_1,\ldots,b_m;1^{t_1}) \leq \widetilde{\mathrm{K}}(x_1,\ldots,x_m;1^t)+s+O(\log(ts)),$$

where $m$ and $t_1$ are parameters chosen later.

We claim the correctness of $V$. Assume that there exists a $t$-time program $M$ of size $s$ such that $M(x)=b$ for every $(x,b)\in\mathrm{supp}(\mathcal{E})$. In this case, the string $(x_1,\ldots,x_m,b_1,\ldots,b_m)$ can be described by using $M$ and a program that prints $(x_1,\ldots,x_m)$; thus, we obtain

$$\mathrm{K}^{\mathsf{poly}(t,m)}(x_1,\ldots,x_m,b_1,\ldots,b_m) \leq \mathrm{K}^{p(t)}(x_1,\ldots,x_m)+s+O(\log s).$$

It follows that

$$\begin{aligned}
&\widetilde{\mathrm{K}}\big(x_1,\ldots,x_m,b_1,\ldots,b_m;1^{t_1}\big)\\
\leq{}&\mathrm{K}^{t_1}(x_1,\ldots,x_m,b_1,\ldots,b_m)\\
\leq{}&\mathrm{K}^{p(t)}(x_1,\ldots,x_m)+s+O(\log s)\\
\leq{}&\widetilde{\mathrm{K}}\big(x_1,\ldots,x_m;1^t\big)+O(\log t)+s+O(\log s).
\end{aligned}$$

Here, the second inequality holds by choosing $t_1=\mathsf{poly}(t,m)$. Thus, $V$ accepts with probability 1.

Next, consider uniform random bits $u_1,\ldots,u_m \sim \{0,1\}$. By symmetry of information for Kolmogorov complexity [ZL70], we have

$$\begin{aligned}
&\mathrm{K}(x_1,\ldots,x_m,u_1,\ldots,u_m)\\
\geq{}&\mathrm{K}(x_1,\ldots,x_m)+\mathrm{K}(u_1,\ldots,u_m \mid x_1,\ldots,x_m)-O(\log nm)\\
\geq{}&\mathrm{K}(x_1,\ldots,x_m)+m-O(\log nm),
\end{aligned}$$

where the last inequality holds with probability at least $1-o(1)$ by Fact 4.2. By Markov's inequality, we also have

$$\Pr_{\forall i,\,(x_i,b_i)\sim\mathcal{E}}\left[\mathrm{K}^t(x_1,\ldots,x_m)-\mathrm{K}(x_1,\ldots,x_m)\geq 2\mathrm{sd}^t_m(\mathcal{E})\right]\leq\frac{1}{2}.$$

Thus, with probability at least $\frac{1}{2} - o(1)$, it holds that

$$\begin{aligned}
&\widetilde{\mathrm{K}}(x_1, \dots, x_m, u_1, \dots, u_m; 1^{t_1}) - \widetilde{\mathrm{K}}(x_1, \dots, x_m; 1^t) \\
\geq\, &\mathrm{K}(x_1, \dots, x_m, u_1, \dots, u_m) - \log p(t_1) - \mathrm{K}^t(x_1, \dots, x_m) \\
\geq\, &m - 2\mathrm{sd}^t_m(\mathcal{E}) - \log p(t_1) - O(\log nm). \\
\geq\, &s + O(\log st). \qquad (13)
\end{aligned}$$

We now assume that the randomized algorithm $V$ outputs 1 with probability at least $\frac{3}{4}$ on input $(\mathcal{E}, 1^t, 1^s)$. We claim that the instance $(\mathcal{E}, 1^t, 1^s)$ is not a No instance. Let $D$ be a program that, on input $(x_1, \dots, x_m, b_1, \dots, b_m)$, outputs 1 if and only if

$$\widetilde{\mathrm{K}}(x_1, \dots, x_m, b_1, \dots, b_m; 1^{t_1}) - \widetilde{\mathrm{K}}(x_1, \dots, x_m; 1^t) \leq s + O(\log(ts)).$$

On one hand, by the assumption, we have

$$\Pr_{\forall i,\, (x_i, b_i) \sim \mathcal{E}}[D(x_1, \dots, x_m, b_1, \dots, b_m) = 1] \geq \frac{3}{4}.$$

On the other hand, by Eq. (13), we have

$$\Pr[D(x_1, \dots, x_m, u_1, \dots, u_m) = 1] \leq \frac{1}{2} + o(1).$$

Using Yao's next-bit predictor, there exists a $D$-oracle program $M^D$ such that

$$\Pr_{(x,b) \sim \mathcal{E}}\left[M^D(x) = b\right] \geq \frac{1}{2} + \frac{1}{5m}.$$

Here, $D$ is polynomial-time computable, so we get a polynomial-time program $M' := M^D$. The program $M'$ takes $x_1, \dots, x_m, b_1, \dots, b_i, u_{i+1} \dots, u_m$ as hard-wired input for some $i$. Thus, the size of $M'$ is at most $mn + m + O(\log ts)$. This shows that the instance $(\mathcal{E}, 1^t, 1^s)$ is not a No instance. □

We now justify the assumption that $\mathrm{sd}^t_m(\mathcal{E})$ is small. We show that any efficient procedure cannot produce a distribution with large sampling depth.

**Lemma A.4.** *Assume* DistNP $\subseteq$ AvgP. *Let $R$ be a randomized polynomial-time reduction. Then, there exists a polynomial $t$ such that for every input $\varphi \in \{0,1\}^*$, with probability at least $\frac{3}{4}$ over the internal randomness of $R$, the reduction $R(\varphi)$ outputs $(\mathcal{E}, s)$ such that*

$$\mathrm{sd}^{t(|x|)}_m(\mathcal{E}) \leq O(|\varphi|),$$

*where $m = m(s, |\varphi|)$ is an arbitrary function.*

*Proof Sketch.* Hirahara and Nanashima [HN21] showed that under the assumption that DistNP $\subseteq$ AvgP, any samplable distribution with $a$ bits of advice has sampling depth at most $O(a)$. Consider the following sampling procedure: Let $\mathcal{E} := R(\varphi)$, sample $x_1, \dots, x_m \sim \mathcal{E}$, and output $x_1, \dots, x_m$. Since the sampling depth of this procedure is at most $O(|\varphi|)$, by Markov's inequality, we obtain that with high probability over the internal randomness of $R$, the sampling depth of $\mathcal{E}$ is at most $O(|\varphi|)$. □

Proposition A.1 follows from the two lemmas above.

## B MINLT and Ko's Relativization Barrier

In this section, we extend Ko's relativization barrier to randomized reductions. Along the way, we also clarify the relationship between MINLT and our learning problem.

**Proposition B.1.** *Let $t$ be any polynomial and $\epsilon \in (0,1)$ be any constant. Then, there exists an oracle $A$ such that $\mathrm{Gap}_{t,2,\epsilon}\mathrm{Learn}^A \in \mathsf{BPP}^A$ and $\mathsf{NP}^A \not\subseteq \mathsf{BPP}^A$.*

This result shows that Theorem 7.2 is non-relativizing. To prove it, we first observe that GapLearn is reducible to MINLT, which is formally defined as follows.

**Definition B.2** ([Ko91])**.** *For a function $g\colon \mathbb{N} \to \mathbb{N}$ and an oracle $A$, the problem $\mathrm{Gap}_g\mathrm{MINLT}^A$ is defined as follows: The input consists of $1^t, 1^s$, and the set $S = \{(x_1, b_1), \dots, (x_m, b_m)\}$ of samples, where $(x_i, b_i) \in \{0,1\}^n \times \{0,1\}$ for some $n \in \mathbb{N}$. We say that an $A$-oracle program $M^A$ is* consistent *with $S$ if $M^A(x) = b$ for every $(x, b) \in S$. The task is to distinguish the following cases:*

**Yes:** *There exists an $A$-oracle $t$-time program of size $s$ consistent with $S$.*

**No:** *No $A$-oracle program of size $s + g(t)$ is consistent with $S$.*[24]

*We define $\mathrm{MINLT}^A$ to be the language consisting of $(S, 1^t, 1^s)$ such that there exists an $A$-oracle $t$-time program of size $s$ consistent with $S$.*

**Fact B.3.** *Let $t, \epsilon$ be as in Proposition B.1. Let $g$ be any function such that $g(t) = t^{o(1)}$. Then, $\mathrm{Gap}_{t,2,\epsilon}\mathrm{Learn}$ is reducible to $\mathrm{Gap}_g\mathrm{MINLT}$ under polynomial-time randomized reductions. Moreover, this statement can be relativized.*

*Proof.* Consider a reduction $R$ that operates as follows: Let $m$ be a parameter chosen later. Let $(\mathcal{E}, 1^s)$ be an instance of GapLearn such that $\mathrm{supp}(\mathcal{E}) \subseteq \{0,1\}^n \times \{0,1\}$. The reduction $R$ picks $m$ samples $(x_1, b_1), \dots, (x_m, b_m) \sim \mathcal{E}$ independently, and outputs $S = \{(x_1, b_1), \dots, (x_m, b_m)\}$ and parameters $(1^{t(n)}, 1^s)$ as an instance of GapMINLT.

The reduction clearly maps a Yes instance of GapLearn to a Yes instance of GapMINLT. To see that a No instance $(\mathcal{E}, 1^s)$ of GapLearn is mapped to a No instance of MINLT with high probability, consider any program $M$ of size $s + g(t(n))$. By the definition of $\mathrm{Gap}_{t,2,\epsilon}\mathrm{Learn}$, we have $s \ge n^{\Omega(1)} \ge t(n)^{\Omega(1)} \ge g(t(n))$ and thus $s + g(t(n)) \le 2s$. The probability that $M(x) = b$ over a random choice of $(x, b) \sim \mathcal{E}$ is at most $\frac{1}{2} + \frac{\epsilon}{2} =: 1 - \delta$. Thus, the probability that $M(x_i) = b_i$ for all $i \in [m]$ is at most

$$(1 - \delta)^m \le \exp(-\delta m) \le 2^{-2s-2},$$

where the last inequality holds by choosing a large $m = O(s/\delta)$. By taking a union bound over all programs of size $2s$, the probability that there exists a program $M$ of size $s + g(t(n)) \le 2s$ such that $M(x_i) = b_i$ for all $i \in [m]$ is at most $\frac{1}{4}$. Thus, with probability at least $\frac{3}{4}$, the instance $S$ is a No instance of $\mathrm{Gap}_g\mathrm{MINLT}$. □

**Lemma B.4.** *There exists an oracle $\mathcal{O}$ such that $\mathrm{Gap}_g\mathrm{MINLT}^{\mathcal{O}} \in \mathsf{P}^{\mathcal{O}}$ and $\mathsf{NP}^{\mathcal{O}} \not\subseteq \mathsf{BPP}^{\mathcal{O}}$, where $g(t) := \log^4 t$.*

Observe that Proposition B.1 immediately follows from Fact B.3 and Lemma B.4. It remains to prove Lemma B.4.

[24] For simplicity, we do not impose any time bound in the No case.

*Proof of Lemma B.4.* For an oracle $A \subseteq \{0,1\}^*$ and a bit $b \in \{0,1\}$, let $bA$ denote $\{bx \mid x \in A\}$. We construct two oracles $A \subseteq \{0,1\}^*$ and $B \subseteq \{0,1\}^*$. The final oracle $\mathcal{O}$ is defined to be $A + B := 0A \cup 1B$. The idea of the oracle construction is as follows: We construct an oracle $A$ that encodes $\mathrm{MINLT}^A$, which ensures $\mathrm{MINLT}^A \in \mathsf{P}^A$. Then we add a "sparse" oracle $B$ so that $\mathsf{NP}^{A+B} \not\subseteq \mathsf{BPP}^{A+B}$. Finally, we argue that $\mathrm{MINLT}^A \in \mathsf{P}^A$ implies $\mathrm{GapMINLT}^{A+B} \in \mathsf{P}^{A+B}$.

We construct $A$ so that $\mathrm{GapMINLT}^{\mathcal{O}}$ is easy. Let $f\colon \{0,1\}^* \to \{0,1\}^*$ be the function that maps $x = (S, 1^t, 1^s)$ to $f(x) := x01^t$. We define $A$ so that

$$x \in \mathrm{MINLT}^A \iff f(x) \in A.$$

This is well defined: Since any $t$-time $A$-oracle program cannot query a string of length $> t$, whether $x \in \mathrm{MINLT}^A$ or not is determined by $\{x \in A \mid |x| \le t\}$; in particular, it does not depend on $f(x)$ because $|f(x)| > t$. Thus, such an oracle $A$ can be inductively defined.

Now, we construct an oracle $B$ so that no $\mathsf{BPP}^{\mathcal{O}}$ algorithm can decide $\mathsf{NP}^{\mathcal{O}}$. For every oracle $B$, let $L_B := \{1^n \mid \{0,1\}^n \cap B \ne \emptyset\}$. Clearly, $L_B \in \mathsf{NP}^{\mathcal{O}}$. Enumerate all the randomized polynomial-time machines $M_0, M_1, M_2, \ldots$, where $M_e$ runs in time $n^{\log n}$ on inputs of length $n$. We construct an oracle $B$ in stages; at stage $e$, we construct $B_{e+1}$; the final oracle $B$ is defined to be $\bigcup_{e \in \mathbb{N}} B_{e+1}$. Let $B_0 := \emptyset$, $n_0 := 2$. At stage $e \in \mathbb{N}$, let $n = n_{e+1}$ be a power of 2 such that $n > (n_e)^{\log n_e}$ and $4n^{\log n} < 2^n$. We consider two cases.

1. If $\Pr\big[M_e^{A+B_e}(1^n) = 1\big] \ge \frac{1}{2}$, then let $B_{e+1} := B_e$ and go to the next stage. In the later stages, we do not add any string of length less than $n^{\log n}$ to $B_e$; thus, we have

$$\Pr\big[M_e^{A+B}(1^n) \ne L_B(1^n)\big] = \Pr\big[M_e^{A+B_e}(1^n) \ne 0\big] \ge \frac{1}{2},$$

   which implies that $M_e^{A+B}$ does not decide $L_B$ on input $1^n$.

2. Otherwise, let $x \in \{0,1\}^n$ be the lexicographically first string such that the probability that $x$ is queried by the $B_e$-oracle machine $M_e^{A+B_e}$ on input $1^n$ is at most $\frac{1}{4}$. Since the number of strings that can be queried by $M_e$ is at most $n^{\log n}$, there exists such a string $x$ because $4n^{\log n} < 2^n$. Then, we define $B_{e+1} := B_e \cup \{x\}$. Note that

$$\Pr\big[M_e^{A+B}(1^n) = L_B(1^n)\big] = \Pr\big[M_e^{A+B_{e+1}}(1^n) = 1\big] \le \frac{1}{2} + \frac{1}{4} = \frac{3}{4},$$

   which implies that $M_e^{A+B}$ does not decide $L_B$ on input $1^n$.

This completes the construction of $B$. By the construction, we have $L_B \notin \mathsf{BPP}^{A+B}$. We also note that $B$ is sparse: for every $n \in \mathbb{N}$, the number of strings $x$ such that $x \in B$ and $|x| \le n$ is at most $O(\log n)$. Moreover, any string $x \in B$ of length $n$ satisfies $\mathrm{K}^{O(n)}(x) \le \log(4n^{\log n} + 1) + O(1) \le O(\log^2 n)$. In particular, $\{x \in B \mid |x| \le n\}$ can be efficiently described by $O(\log n) \cdot O(\log^2 n)$ bits.

It remains to prove that $\mathrm{GapMINLT}^{A+B} \in \mathsf{P}^{A+B}$. Consider an $(A+B)$-oracle algorithm $V^{A+B}$ that accepts an instance $(S, 1^t, 1^s)$ if and only if $f(S, 1^{t'}, 1^{s+O(\log^3 t)}) \in A$, where $t' = \mathsf{poly}(t)$ is a parameter chosen later. We claim that $V^{A+B}$ decides $\mathrm{GapMINLT}^{A+B}$. Consider a Yes instance $(S, 1^t, 1^s)$ of $\mathrm{GapMINLT}^{A+B}$. Let $M^{A+B}$ be a $t$-time program of size $s$ consistent with $S$. Since the set $\{q \in B \mid |q| \le t\}$ can be efficiently described using $O(\log^3 t)$ bits, the program $M^{A+B}$ can be simulated by an $A$-oracle $t'$-time program of size $O(\log^3 t)$ for some $t' = \mathsf{poly}(t)$. We obtain $(S, 1^{t'}, 1^{s+O(\log^3 t)}) \in \mathrm{MINLT}^A$, which implies that $f(S, 1^{t'}, 1^{s+O(\log^3 t)}) \in A$, as desired. Next, consider a No instance $(S, 1^t, 1^s)$ of GapMINLT$^{A+B}$. In this case, there exists no $(A+B)$-oracle program $M^{A+B}$ of size $s+g(t)$ consistent with $S$. In particular, there exists no $A$-oracle program $M^A$ of size $s+g(t)-O(1)$ consistent with $S$. Since $s+g(t)-O(1) \geq s+O(\log^3 t)$, we obtain $(S, 1^{t'}, 1^{s+O(\log^3 t)}) \notin \text{MINLT}^A$. ☐

## C NP-Hardness of Learning Small Circuits

In this appendix, we mention a folklore result of NP-hardness of properly learning small circuits [HJLT96; ABFKP08; ILO20].

**Proposition C.1.** *Given a distribution $\mathcal{D}$ represented by a circuit and a size parameter $s \in \mathbb{N}$, it is* NP*-hard to distinguish the following cases:*

**Yes:** *There exists a circuit $C$ of size $s$ such that* $\Pr_{(x,b)\sim\mathcal{D}}[C(x) = b] = 1$.

**No:** *For any circuit $C$ of size $s$,* $\Pr_{(x,b)\sim\mathcal{D}}[C(x) = b] < 1$.

The result easily follows from NP-hardness of learning $s$-juntas [HJLT96; ABFKP08]. For completeness, we present a proof.

*Proof.* We reduce the hitting set problem (which is equivalent to the set cover problem). The input of the hitting set problem consists of a family $\mathcal{S} \subseteq 2^{[n]}$ of subsets of $[n]$. The goal is to find a minimum set $H \subseteq [n]$ such that $H \cap S \neq \emptyset$ for every $S \in \mathcal{S}$. We reduce the instance $\mathcal{S}$ to a distribution $\mathcal{D}$ such that $\text{supp}(\mathcal{D})$ consists of $(0^n, 0)$ and $(\chi_S, 1)$ for every $S \in \mathcal{S}$, where $\chi_S \in \{0,1\}^n$ is the characteristic vector of $S$. We claim that there exists a hitting set $H$ of size $s$ if and only if there exists a circuit of size $s-1$ such that $C(x) = b$ for every $(C, b) \in \text{supp}(\mathcal{D})$. If there exists a hitting set $H = \{h_1, \ldots, h_s\} \subseteq [n]$ of size $s$, then the circuit $x_{h_1} \vee \cdots \vee x_{h_s}$ is a circuit of size $s-1$ that is consistent with $\text{supp}(\mathcal{D})$. Here, $x_1, \ldots, x_n$ denote the input variables of $C$, and the size of a circuit is measured by the number of AND and OR gates. Conversely, if there exists a circuit $C$ of size $s-1$ that is consistent with $\text{supp}(\mathcal{D})$, then the circuit $C$ is an $s$-junta, i.e., depends on at most $s$ inputs. Since $C(0^n) = 0$ and $C(\chi_S) = 1$ for every $S \in \mathcal{S}$, the circuit $C$ must depend on $x_i$ for some $i \in S$. Thus, the set of the input variables on which $C$ depends is a hitting set of size $s$. ☐

Although this folklore result might look superficially similar to Theorem 1.1, there are significant differences:

1. The size $s$ of an optimal circuit must be always smaller than the input length $|x|$. In particular, it is not possible to make $|x| = O(\log n)$ as in Theorem 1.2 and prove NP-hardness of MCSP$^*$.
2. It is unknown to us whether the relativization barrier of Appendix B is applicable to circuits. Moreover, the definition of $A$-oracle circuits may be controversial; the size of $A$-oracle circuits can be defined as the number of wires or the number of gates, etc.
3. The hardness of approximation is weak, as the hitting set problem can be approximated within a factor of $O(\log n)$ by a greedy algorithm.
4. The results of Appendix A would be significantly weaker for circuits.

## Acknowledgments

This work was supported by JST, PRESTO Grant Number JPMJPR2024, Japan. I thank Leonid Levin for clarifying that the second problem in [Lev73] is DNF-MCSP* and anonymous referees for helpful comments.

## References

[ABFKP08] Michael Alekhnovich, Mark Braverman, Vitaly Feldman, Adam R. Klivans, and Toniann Pitassi. "The complexity of properly learning simple concept classes". In: *J. Comput. Syst. Sci.* 74.1 (2008), pp. 16–34. DOI: 10.1016/j.jcss.2007.04.011.

[ABKMR06] Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. "Power from Random Strings". In: *SIAM J. Comput.* 35.6 (2006), pp. 1467–1493. DOI: 10.1137/050628994.

[ABMP01] Michael Alekhnovich, Samuel R. Buss, Shlomo Moran, and Toniann Pitassi. "Minimum Propositional Proof Length Is NP-Hard to Linearly Approximate". In: *J. Symb. Log.* 66.1 (2001), pp. 171–191. DOI: 10.2307/2694916.

[ABX08] Benny Applebaum, Boaz Barak, and David Xiao. "On Basing Lower-Bounds for Learning on Worst-Case Assumptions". In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 2008, pp. 211–220. DOI: 10.1109/FOCS.2008.35.

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

[AGGM06] Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. "On basing one-way functions on NP-hardness". In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 2006, pp. 701–710. DOI: 10.1145/1132516.1132614.

[AH19] Eric Allender and Shuichi Hirahara. "New Insights on the (Non-)Hardness of Circuit Minimization and Related Problems". In: *TOCT* 11.4 (2019), 27:1–27:27. DOI: 10.1145/3349616.

[AHMPS08] Eric Allender, Lisa Hellerstein, Paul McCabe, Toniann Pitassi, and Michael E. Saks. "Minimizing Disjunctive Normal Form Formulas and $AC^0$ Circuits Given a Truth Table". In: *SIAM J. Comput.* 38.1 (2008), pp. 63–84. DOI: 10.1137/060664537.

[AKRR11] Eric Allender, Michal Koucký, Detlef Ronneburger, and Sambuddha Roy. "The pervasive reach of resource-bounded Kolmogorov complexity in computational complexity theory". In: *J. Comput. Syst. Sci.* 77.1 (2011), pp. 14–40. DOI: 10.1016/j.jcss.2010.06.004.

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

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

[BB15] Andrej Bogdanov and Christina Brzuska. "On Basing Size-Verifiable One-Way Functions on NP-Hardness". In: *Proceedings of the Theory of Cryptography Conference (TCC)*. 2015, pp. 1–6. DOI: 10.1007/978-3-662-46494-6_1.

[BEHW87] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. "Occam's Razor". In: *Inf. Process. Lett.* 24.6 (1987), pp. 377–380. DOI: 10.1016/0020-0190(87)90114-1.

[Bei11] Amos Beimel. "Secret-Sharing Schemes: A Survey". In: *The Third International Workshop on Coding and Cryptology (IWCC)*. 2011, pp. 11–46. DOI: 10.1007/978-3-642-20901-7_2.

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

[BFT98] Harry Buhrman, Lance Fortnow, and Thomas Thierauf. "Nonrelativizing Separations". In: *Proceedings of the Conference on Computational Complexity (CCC)*. 1998, pp. 8–12. DOI: 10.1109/CCC.1998.694585.

[BGGS16] Arnab Bhattacharyya, Ameet Gadekar, Suprovat Ghoshal, and Rishi Saket. "On the Hardness of Learning Sparse Parities". In: *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*. 2016, 11:1–11:17. DOI: 10.4230/LIPIcs.ESA.2016.11.

[BGLR94] Mihir Bellare, Shafi Goldwasser, Carsten Lund, and Alexander Russell. "Efficient probabilistic checkable proofs and applications to approximation". In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 1994, p. 820. DOI: 10.1145/195058.195467.

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

[BHPT20] Amey Bhangale, Prahladh Harsha, Orr Paradise, and Avishay Tal. "Rigid Matrices From Rectangular PCPs". In: *Electron. Colloquium Comput. Complex.* (2020), p. 75.

[BL88] Josh Cohen Benaloh and Jerry Leichter. "Generalized Secret Sharing and Monotone Functions". In: *Proceedings of the International Cryptology Conference (CRYPTO)*. 1988, pp. 27–35. DOI: 10.1007/0-387-34799-2_3.

[Bla79] George Robert Blakley. "Safeguarding cryptographic keys". In: *International Workshop on Managing Requirements Knowledge (MARK)*. 1979, pp. 313–318. DOI: 10.1109/MARK.1979.8817296.

[BP92] Raymond A. Board and Leonard Pitt. "On the Necessity of Occam Algorithms". In: *Theor. Comput. Sci.* 100.1 (1992), pp. 157–184. DOI: 10.1016/0304-3975(92)90367-O.

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

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

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

[CIKK16] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. "Learning Algorithms from Natural Proofs". In: *Proceedings of the Conference on Computational Complexity (CCC)*. 2016, 10:1–10:24. DOI: 10.4230/LIPIcs.CCC.2016.10.

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

[DFKRS11] Irit Dinur, Eldar Fischer, Guy Kindler, Ran Raz, and Shmuel Safra. "PCP Characterizations of NP: Toward a Polynomially-Small Error-Probability". In: *Comput. Complex.* 20.3 (2011), pp. 413–504. DOI: 10.1007/s00037-011-0014-4.

[DHK15] Irit Dinur, Prahladh Harsha, and Guy Kindler. "Polynomially Low Error PCPs with polyloglog n Queries via Modular Composition". In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 2015, pp. 267–276. DOI: 10.1145/2746539.2746630.

[DLS14] Amit Daniely, Nati Linial, and Shai Shalev-Shwartz. "From average case complexity to improper learning complexity". In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 2014, pp. 441–448. DOI: 10.1145/2591796.2591820.

[DS04] Irit Dinur and Shmuel Safra. "On the hardness of approximating label-cover". In: *Inf. Process. Lett.* 89.5 (2004), pp. 247–254. DOI: 10.1016/j.ipl.2003.11.007.

[DS16] Amit Daniely and Shai Shalev-Shwartz. "Complexity Theoretic Limitations on Learning DNF's". In: *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*. 2016, pp. 815–830.

[DV21] Amit Daniely and Gal Vardi. "From Local Pseudorandom Generators to Hardness of Learning". In: *Conference on Learning Theory, COLT 2021, 15-19 August 2021, Boulder, Colorado, USA*. 2021, pp. 1358–1394.

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

[FGKP09] Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. "On Agnostic Learning of Parities, Monomials, and Halfspaces". In: *SIAM J. Comput.* 39.2 (2009), pp. 606–645. DOI: 10.1137/070684914.

[FGRW12] Vitaly Feldman, Venkatesan Guruswami, Prasad Raghavendra, and Yi Wu. "Agnostic Learning of Monomials by Halfspaces Is Hard". In: *SIAM J. Comput.* 41.6 (2012), pp. 1558–1590. DOI: 10.1137/120865094.

[FLV06] Lance Fortnow, Troy Lee, and Nikolai K. Vereshchagin. "Kolmogorov Complexity with Error". In: *STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science, Marseille, France, February 23-25, 2006, Proceedings*. 2006, pp. 137–148. DOI: 10.1007/11672142_10.

[GKS10] Parikshit Gopalan, Subhash Khot, and Rishi Saket. "Hardness of Reconstructing Multivariate Polynomials over Finite Fields". In: *SIAM J. Comput.* 39.6 (2010), pp. 2598–2621. DOI: 10.1137/070705258.

[GL89] Oded Goldreich and Leonid A. Levin. "A Hard-Core Predicate for all One-Way Functions". In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 1989, pp. 25–32. DOI: 10.1145/73007.73010.

[Gol11] Oded Goldreich. "A Sample of Samplers: A Computational Perspective on Sampling". In: *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*. Springer, 2011, pp. 302–332. DOI: 10.1007/978-3-642-22670-0_24.

[Hir18] Shuichi Hirahara. "Non-Black-Box Worst-Case to Average-Case Reductions within NP". In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 2018, pp. 247–258. DOI: 10.1109/FOCS.2018.00032.

[Hir20a] Shuichi Hirahara. "Non-Disjoint Promise Problems from Meta-Computational View of Pseudorandom Generator Constructions". In: *Proceedings of the Computational Complexity Conference (CCC)*. 2020, 20:1–20:47. DOI: 10.4230/LIPIcs.CCC.2020.20.

[Hir20b] Shuichi Hirahara. "Unexpected hardness results for Kolmogorov complexity under uniform reductions". In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 2020, pp. 1038–1051. DOI: 10.1145/3357713.3384251.

[Hir21] Shuichi Hirahara. "Average-case hardness of NP from exponential worst-case hardness assumptions". In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 2021, pp. 292–302. DOI: 10.1145/3406325.3451065.

[Hir22] Shuichi Hirahara. "Symmetry of Information from Meta-Complexity". In: *Proceedings of the Computational Complexity Conference (CCC)*. 2022, 26:1–26:41. DOI: 10.4230/LIPIcs.CCC.2022.26.

[HJLT96] Thomas R. Hancock, Tao Jiang, Ming Li, and John Tromp. "Lower Bounds on Learning Decision Lists and Trees". In: *Inf. Comput.* 126.2 (1996), pp. 114–122. DOI: 10.1006/inco.1996.0040.

[HN21] Shuichi Hirahara and Mikito Nanashima. "On Worst-Case Learning in Relativized Heuristica". In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 2021, pp. 751–758. DOI: 10.1109/FOCS52979.2021.00078.

[HOS18] Shuichi Hirahara, Igor Carboni Oliveira, and Rahul Santhanam. "NP-hardness of Minimum Circuit Size Problem for OR-AND-MOD Circuits". In: *Proceedings of the Computational Complexity Conference (CCC)*. 2018, 5:1–5:31. DOI: 10.4230/LIPIcs.CCC.2018.5.

[HP15] John M. Hitchcock and Aduri Pavan. "On the NP-Completeness of the Minimum Circuit Size Problem". In: *Proceedings of the Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS)*. 2015, pp. 236–245. DOI: 10.4230/LIPIcs.FSTTCS.2015.236.

[HS17] Shuichi Hirahara and Rahul Santhanam. “On the Average-Case Complexity of MCSP and Its Variants”. In: *Proceedings of the Computational Complexity Conference (CCC)*. 2017, 7:1–7:20. DOI: 10.4230/LIPIcs.CCC.2017.7.

[HVV06] Alexander Healy, Salil P. Vadhan, and Emanuele Viola. “Using Nondeterminism to Amplify Hardness”. In: *SIAM J. Comput.* 35.4 (2006), pp. 903–931. DOI: 10.1137/S0097539705447281.

[HW16] Shuichi Hirahara and Osamu Watanabe. “Limits of Minimum Circuit Size Problem as Oracle”. In: *Proceedings of the Conference on Computational Complexity (CCC)*. 2016, 18:1–18:20. DOI: 10.4230/LIPIcs.CCC.2016.18.

[HW20] Shuichi Hirahara and Osamu Watanabe. “On Nonadaptive Security Reductions of Hitting Set Generators”. In: *Proceedings of the Approximation, Randomization, and Combinatorial Optimization (APPROX/RANDOM)*. 2020, 15:1–15:14. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2020.15.

[IL90] Russell Impagliazzo and Leonid A. Levin. “No Better Ways to Generate Hard NP Instances than Picking Uniformly at Random”. In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 1990, pp. 812–821. DOI: 10.1109/FSCS.1990.89604.

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

[Ila20b] Rahul Ilango. “Constant Depth Formula and Partial Function Versions of MCSP are Hard”. In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 2020, pp. 424–433.

[Ila21] Rahul Ilango. “The Minimum Formula Size Problem is (ETH) Hard”. In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 2021, pp. 427–432. DOI: 10.1109/FOCS52979.2021.00050.

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

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

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

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. “Which Problems Have Strongly Exponential Complexity?” In: *J. Comput. Syst. Sci.* 63.4 (2001), pp. 512–530. DOI: 10.1006/jcss.2001.1774.

[ISN93] Mitsuru Ito, Akira Saio, and Takao Nishizeki. “Multiple Assignment Scheme for Sharing Secret”. In: *J. Cryptol.* 6.1 (1993), pp. 15–20. DOI: 10.1007/BF02620229.

[ISW06] Russell Impagliazzo, Ronen Shaltiel, and Avi Wigderson. "Reducing The Seed Length In The Nisan-Wigderson Generator". In: *Combinatorica* 26.6 (2006), pp. 647–681. DOI: 10.1007/s00493-006-0036-8.

[IW97] Russell Impagliazzo and Avi Wigderson. "$P = BPP$ if $E$ Requires Exponential Circuits: Derandomizing the XOR Lemma". In: *Proceedings of the Symposium on the Theory of Computing (STOC)*. 1997, pp. 220–229. DOI: 10.1145/258533.258590.

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

[KMS12] Jeff Kinne, Dieter van Melkebeek, and Ronen Shaltiel. "Pseudorandom Generators, Typically-Correct Derandomization, and Circuit Lower Bounds". In: *Computational Complexity* 21.1 (2012), pp. 3–61. DOI: 10.1007/s00037-011-0019-z.

[KMV08] Adam Tauman Kalai, Yishay Mansour, and Elad Verbin. "On agnostic boosting and parity learning". In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 2008, pp. 629–638. DOI: 10.1145/1374376.1374466.

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

[Kol65] Andrei N Kolmogorov. "Three approaches to the quantitative definition of information". In: *Problems of information transmission* 1.1 (1965), pp. 1–7.

[KS08] Subhash Khot and Rishi Saket. "Hardness of Minimizing and Learning DNF Expressions". In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 2008, pp. 231–240. DOI: 10.1109/FOCS.2008.37.

[KS11] Subhash Khot and Rishi Saket. "On the hardness of learning intersections of two halfspaces". In: *J. Comput. Syst. Sci.* 77.1 (2011), pp. 129–141. DOI: 10.1016/j.jcss.2010.06.010.

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

[LMN93] Nathan Linial, Yishay Mansour, and Noam Nisan. "Constant Depth Circuits, Fourier Transform, and Learnability". In: *J. ACM* 40.3 (1993), pp. 607–620. DOI: 10.1145/174130.174138.

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

[LP22] Yanyi Liu and Rafael Pass. "On One-Way Functions from NP-Complete Problems". In: *Proceedings of the Computational Complexity Conference (CCC)*. 2022, 36:1–36:24. DOI: 10.4230/LIPIcs.CCC.2022.36.

[Mas79] William J Masek. "Some NP-complete set covering problems". In: *Unpublished manuscript* (1979).

[MW17] Cody D. Murray and R. Ryan Williams. "On the (Non) NP-Hardness of Computing Circuit Complexity". In: *Theory of Computing* 13.1 (2017), pp. 1–22. DOI: 10.4086/toc.2017.v013a004.

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

[OS18] Igor Carboni Oliveira and Rahul Santhanam. “Hardness Magnification for Natural Problems”. In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 2018, pp. 65–76.

[PV88] Leonard Pitt and Leslie G. Valiant. “Computational limitations on learning from examples”. In: *J. ACM* 35.4 (1988), pp. 965–984. DOI: 10.1145/48014.63140.

[San20] Rahul Santhanam. “Pseudorandomness and the Minimum Circuit Size Problem”. In: *Proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*. 2020, 68:1–68:26. DOI: 10.4230/LIPIcs.ITCS.2020.68.

[Sch90] Robert E. Schapire. “The Strength of Weak Learnability”. In: *Mach. Learn.* 5 (1990), pp. 197–227. DOI: 10.1007/BF00116037.

[Sha79] Adi Shamir. “How to Share a Secret”. In: *Commun. ACM* 22.11 (1979), pp. 612–613. DOI: 10.1145/359168.359176.

[SS20] Michael Saks and Rahul Santhanam. “Circuit Lower Bounds from NP-Hardness of MCSP Under Turing Reductions”. In: *Proceedings of the Computational Complexity Conference (CCC)*. 2020, 26:1–26:13. DOI: 10.4230/LIPIcs.CCC.2020.26.

[Tra84] Boris A. Trakhtenbrot. “A Survey of Russian Approaches to Perebor (Brute-Force Searches) Algorithms”. In: *IEEE Annals of the History of Computing* 6.4 (1984), pp. 384–400. DOI: 10.1109/MAHC.1984.10036.

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

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

[Uhl74] Dietmar Uhlig. “On the synthesis of self-correcting schemes from functional elements with a small number of reliable elements”. In: *Mathematical Notes of the Academy of Sciences of the USSR* 15.6 (1974), pp. 558–562.

[Uhl92] Dietmar Uhlig. “Networks Computing Boolean Functions for Multiple Input Values”. In: *Poceedings of the London Mathematical Society Symposium on Boolean Function Complexity*. London, United Kingdom: Cambridge University Press, 1992, pp. 165–173. ISBN: 0521408261.

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

[Vad17] Salil P. Vadhan. “On Learning vs. Refutation”. In: *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*. 2017, pp. 1835–1848.

[Val84] Leslie G. Valiant. “A Theory of the Learnable”. In: *Commun. ACM* 27.11 (1984), pp. 1134–1142. DOI: 10.1145/1968.1972.

[Weg87] Ingo Wegener. *The complexity of Boolean functions*. Wiley-Teubner, 1987.

[Yao82] Andrew Chi-Chih Yao. “Theory and Applications of Trapdoor Functions (Extended Abstract)”. In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 1982, pp. 80–91. DOI: 10.1109/SFCS.1982.45.

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