Electronic Colloquium on Computational Complexity, Report No. 89 (2021)

A Relativization Perspective on Meta-Complexity

Hanlin Ren*
IIIS, Tsinghua University

Rahul Santhanam†
University of Oxford

June 25, 2021

Abstract

Meta-complexity studies the complexity of computational problems about complexity theory, such as the Minimum Circuit Size Problem (MCSP) and its variants. We show that a relativization barrier applies to many important open questions in meta-complexity. We give relativized worlds where:

1. MCSP can be solved in deterministic polynomial time, but the search version of MCSP cannot be solved in deterministic polynomial time, even approximately. In contrast, Carmosino, Impagliazzo, Kabanets, Kolokolova [CCC'16] gave a randomized approximate search-to-decision reduction for MCSP with a relativizing proof.
2. The complexities of MCSP$[2^{n/2}]$ and MCSP$[2^{n/4}]$ are different, in both worst-case and average-case settings. Thus the complexity of MCSP is not "robust" to the choice of the size function.
3. Levin's time-bounded Kolmogorov complexity $\mathrm{Kt}(x)$ can be approximated to a factor $(2+\epsilon)$ in polynomial time, for any $\epsilon > 0$.
4. Natural proofs do not exist, and neither do auxiliary-input one-way functions. In contrast, Santhanam [ITCS'20] gave a relativizing proof that the non-existence of natural proofs implies the existence of one-way functions under a conjecture about optimal hitting sets.
5. DistNP does not reduce to GapMINKT by a family of "robust" reductions. This presents a technical barrier for solving a question of Hirahara [FOCS'20].

*rhl16@mails.tsinghua.edu.cn
†rahul.santhanam@cs.ox.ac.uk

ISSN 1433-8092

Contents

1 Introduction

Meta-complexity refers to the complexity of computing complexity. A prominent example of a meta-complexity problem is the Minimum Circuit Size Problem (MCSP): Given as input the (length-2^n) truth table of a function $f : \{0,1\}^n \to \{0,1\}$, output the size of the smallest circuit that computes f. MCSP was recognized as a fundamental problem in the Soviet Union since 1950s [Tra84], and has received a lot of attention in the last two decades since the seminal work of Kabanets and Cai [KC00]. Other examples include computing variants of Kolmogorov complexity such as polynomial-time bounded Kolmogorov complexity and Levin's time-bounded Kolmogorov complexity Kt [Ko91, ABK+06]. Questions about the circuit size of Boolean functions are closely related to Kolmogorov complexity and incompressibility, because a circuit is essentially a *compressed representation* of the truth table of the function it computes.

There has been plenty of interplay between meta-complexity and other areas of complexity theory such as average-case complexity [HS17, Hir18, Hir20a, Hir21], cryptography [RR97, LP20, San20, RS21], learning theory [CIKK16, OS17] and pseudorandomness [KC00, ABK+06, OS17, Hir20c].

We highlight a couple of recent breakthrough results. The first gives a non-black-box worst-case to average-case reduction for a problem about Kolmogorov complexity ("GapMINKT") that many believe to be NP-hard.

Theorem 1.1 ([Hir18], building on [CIKK16])**.** *There is a randomized polynomial-time worst-case to average-case reduction for* GapMINKT*.*

The second gives an *equivalence* between the existence of one-way functions and the bounded-error average-case hardness over the uniform distribution of the functional version of MINKT. This result *characterizes* the most fundamental primitive in cryptography by a notion in meta-complexity.

Theorem 1.2 ([LP20])**.** *One-way functions exist if and only if there is a polynomial p such that the $p(n)$-time bounded Kolmogorov complexity of a string x of length n cannot be computed in polynomial time on average, when x is chosen uniformly at random from n-bit strings.*

Results such as these give hope for a rich theory connecting complexity lower bounds, meta-complexity, average-case complexity, learning theory and cryptography, among other fields. However, despite much effort, many basic questions about meta-complexity remain elusive. In addition, the recent advances on meta-complexity also propose new questions, some of which are seemingly beyond our reach. (See Section 1.1 for a sample of these questions.)

In this work, we seek a more fine-grained understanding of the current landscape of meta-complexity by using the classical perspective of *relativization* [BGS75]. It is noteworthy that Theorem 1.1 and Theorem 1.2 relativize. Of course, we need to be careful here to define what relativization means, as the notion typically applies to complexity classes and not to computational problems. However, meta-computational problems do indeed have natural notions of relativizations, where the algorithms solving the problem as well as the algorithms defining the problem get access to the same oracle A. Results such as Theorem 1.1 and Theorem 1.2 use techniques from the theory of pseudorandomness [NW94, IW01, TV07], which typically relativize, and it is worth asking how much these techniques can achieve. Can they be used to solve the major open problems in the area?

We give a largely negative answer to this question, by giving oracles relative to which many of the questions in the area have answers opposite to what we expect. However, we do not necessarily infer that there are fundamental barriers to solving the major open questions; we can only say that new techniques will be required in many cases. Our perspective also contributes to formulating new notions and questions which might still be approachable using current techniques. We also note that there are some exciting recent works in meta-complexity by Ilango

and others (e.g. [ILO20, Ila20a, Ila20b, Ila20c]) using gate elimination and related ideas. It is not clear yet whether relativization is a barrier to these techniques.

1.1 Our Questions

We first introduce the questions with which we are concerned.

1.1.1 Easiness or Hardness of Meta-Complexity Problems

Arguably, the most important and fundamental problem about MCSP is whether MCSP is easy or hard. Is MCSP in polynomial time, or if not, is MCSP NP-complete? It is reported in [AKRR11, Lev] that Levin delayed the publication of his NP-completeness results [Lev73] because he wanted to show NP-hardness for MCSP. A long line of research [KC00, MW17, HP15, HW16, AH19, AIV19, SS20, Fu20] showed that the NP-completeness of MCSP implies breakthrough results in complexity theory. For instance, if MCSP is NP-complete under polynomial-time Karp reductions, then EXP $\neq$ ZPP [MW17]. However, these results do not indicate whether MCSP is or is not NP-complete; they merely suggest that this problem will be hard to solve.

Question 1.3. *Is* MCSP NP*-complete under polynomial-time Karp reductions?*

Just as with MCSP, it is open to show the NP-hardness of MINKT. A further motivation for this problem is the recent "non-black-box" worst-case to average-case reduction for MINKT [Hir18]. As a consequence, if GapMINKT is NP-hard, then the worst-case and average-case complexities of NP are equivalent. As there are serious obstacles to showing the NP-completeness of MINKT by "weak" reductions, [Hir18] proposed, as a weakening of Question 1.3, that MINKT could be NP-hard via very powerful reductions:

Question 1.4. *Is* GapMINKT NP*-hard under* $\mathsf{coNP}_{/\mathrm{poly}}$*-Turing reductions?*

In terms of unconditional lower bounds, there is an intriguing question about the meta-complexity of Levin's Kt complexity, raised in [ABK+06]. It is known that MKtP is EXP-complete, but only under rather powerful reductions such as $\mathsf{P}_{/\mathrm{poly}}$-truth-table reductions or NP-Turing reductions. Therefore, it is reasonable to conjecture that MKtP is not in P. However, the aforementioned reducibilities are too strong, so we cannot apply the time hierarchy theorem directly to prove that MKtP $\notin$ P. Still, it may be surprising that this problem has been open for almost 20 years:[1]

Question 1.5. *Is* MKtP *computable (or at least approximable) in polynomial-time?*

(We note that a randomized version of MKtP, called MrKtP, is known to be not in BPP unconditionally [Oli19].)

1.1.2 Structural Properties of Meta-Complexity Problems

Every NP-complete problem admits a *search-to-decision* reduction. For instance, given an oracle that decides SAT, for every input formula φ that is satisfiable, we can find a satisfying assignment of φ in polynomial time. However, it is unknown whether MCSP has this property.

Question 1.6. *Does* MCSP *admit a search-to-decision reduction?*

We remark that there has been some progress on Question 1.6: [CIKK16] showed that if MCSP is in BPP, then a certain "weak" version of search-MCSP can be solved in probabilistic polynomial time; [Ila20b] presented a "non-trivial" search-to-decision reduction for the problem of minimizing formulas.

[1] The conference version of [ABK+06] was published in 2002.

Another mystery about MCSP is whether its various *parameterized* versions are equivalent. Specifically, let MCSP$[s(n)]$ denote the problem that given a truth table of a function $f : \{0,1\}^n \to \{0,1\}$, determine whether f can be computed by a circuit of size $s(n)$. It is easy to see that MCSP$[2^{n/2}]$ reduces to MCSP$[2^{n/4}]$,[2] but the converse direction is unknown:

Question 1.7. *Is* MCSP$[2^{n/4}]$ *reducible to* MCSP$[2^{n/2}]$ *under polynomial-time Karp reductions?*

The average-case version of Question 1.7 is also open. It is observed in [HS17] that any errorless heuristic for MCSP$[2^{n/2}]$ can be transformed into an errorless heuristic for MCSP$[2^{n/4}]$, but the converse is unknown.

Question 1.8. *If* MCSP$[2^{n/4}]$ *is easy on average, does this imply that* MCSP$[2^{n/2}]$ *is also easy on average?*

One drawback of the worst-case to average-case reduction of [Hir18] is that it only works for *zero-error* average-case complexity. Ideally, we would like to establish a worst-case to *two-sided-error* average-case reduction for MINKT. Can we extend the results in [Hir18] to the two-sided-error setting?

Question 1.9. *Is there a natural distribution such that, if* MINKT *is easy on this distribution with two-sided error, then* GapMINKT *is solvable in the worst case? In particular, does the* uniform *distribution satisfy the above condition?*

1.1.3 Meta-Complexity, Average-Case Complexity and Cryptography

Some of the most compelling questions around meta-complexity relate to connections with average-case complexity and cryptography. A partial converse of [Hir18] was established in [Hir20c, Hir20a], where it was shown that if GapMINKT$^{\mathrm{SAT}} \in \mathsf{P}$, then $\mathsf{DistNP} \subseteq \mathsf{AvgP}$, i.e. NP is easy on average. Here GapMINKT$^{\mathrm{SAT}}$ is the problem of determining the (time-bounded) Kolmogorov complexity of a string with a SAT oracle. Based on this result, [Hir20a] characterized the average-case complexity of the polynomial hierarchy by the worst-case complexity of meta-complexity. An important open question, a positive answer to which would imply a characterization of the average-case complexity for NP, is whether the SAT oracle can be removed, that is:

Question 1.10. *Does* GapMINKT $\in \mathsf{P}$ *imply* $\mathsf{DistNP} \subseteq \mathsf{AvgP}$*?*

There seems to be strong correspondences between the hardness of MCSP and problems in cryptography. For example, if MCSP is easy, then one-way functions (OWFs) do not exist [RR97, KC00]. Under the unproven Universality Conjecture, [San20] established the converse direction, i.e. if MCSP is zero-error average-case hard, then OWFs exist. Of course, an *unconditional* answer would be much more interesting:

Question 1.11. *Can we base the existence of OWF from the nonexistence of natural proofs?*

A recent exciting work [LP20] established the equivalence between the two-sided error average-case hardness of MINKT and the existence of one-way functions. Given the result in [LP20], it is perhaps natural to conjecture that GapMINKT $\in \mathsf{CZK}$ unconditionally, where CZK is the set of languages with a computational zero-knowledge proof system [GMW91]. One could imagine a win-win argument as follows: If MINKT is easy, then of course it is in CZK; on the other hand, if MINKT is hard, then one-way functions exist, and by the result of [GMW91], every language in NP is in CZK. However, there are some gaps between the "easy" and "hard" in the above argument, as we do not know what happens if MINKT is only worst-case hard and one-way functions do not exist.

[2] Given an input truth table f of length 2^n, let f' be the concatenation of 2^n copies of f, then $f' : \{0,1\}^{2n} \to \{0,1\}$ is a function that only depends on half of its input bits, and the circuit complexities of f and f' are exactly the same. Therefore $f \in$ MCSP$[2^{n/2}]$ if and only if $f' \in$ MCSP$[2^{n/4}]$.

Question 1.12. *Does (some gap version of)* MCSP *or* MINKT *admit a computational zero knowledge proof system?*

1.2 Our Results

In this work, we investigate the above questions in the perspective of *relativization*.

1.2.1 Meta-Complexity Problems Are Not Robust in Relativized Worlds

In our first set of results, we present evidence for the following hypothesis: A *slight change* in the definition of a meta-complexity problem could result in a *completely different* problem. For example, we show that there are relativized worlds where MCSP is significantly easier than search-MCSP, and relativized worlds where $\mathrm{MCSP}[2^{n/2}]$ and $\mathrm{MCSP}[2^{n/4}]$ have dramatically different complexities.

Theorem 1.13 (Informal version)**.** *For each of the following items, there is a relativized world where it becomes true.*

- $\mathrm{MCSP} \in \mathsf{P}$*, but* search-MCSP *is very hard.*
- $\mathrm{MCSP}[2^{n/2}] \in \mathsf{P}$*, but* $\mathrm{MCSP}[2^{n/4}]$ *is very hard.*
- $\mathrm{MCSP}[2^{n/4}]$ *admits a polynomial-time errorless heuristic, but* $\mathrm{MCSP}[2^{n/2}]$ *does not.*

As direct consequences of Theorem 1.13, we have the following nonreducibility results: For example, unless nonrelativizing techniques are used, MCSP does not admit a search-to-decision reduction, and $\mathrm{MCSP}[2^{n/4}]$ does not reduce to $\mathrm{MCSP}[2^{n/2}]$.

1.2.2 Barriers for Proving Hardness of Levin's Complexity

Our second result concerns Question 1.5.

Theorem 1.14 (Informal version)**.** *There is a relativized world where Levin's* Kt *complexity can be* $(2+\epsilon)$*-approximated in polynomial time.*

We note that Question 1.5 also appeared in a stronger form in literature. In particular, let R_{Kt} be the set of strings x such that $\mathrm{Kt}(x) \geq |x|/3$, it is conjectured that any "dense enough" subset of R_{Kt} is not in polynomial time. Our result shows that this conjecture needs nonrelativizing techniques to prove.

Actually, our message is even stronger than the above statement of Theorem 1.14. We define a nonstandard variant of Levin's Kt complexity, and denote it as $\widetilde{\mathrm{Kt}}$, such that $\widetilde{\mathrm{Kt}}$ approximates Kt, i.e. for every string x, $\widetilde{\mathrm{Kt}}(x) \leq \mathrm{Kt}(x) \leq (2+o(1))\widetilde{\mathrm{Kt}}(x)$. Then we construct a relativized world where $\widetilde{\mathrm{Kt}}$ is computable in polynomial time *exactly*, and Theorem 1.14 follows directly.

However, non-relativizing techniques already play an important role in characterizing the complexity of R_{Kt}. It was shown that any dense subset of R_{Kt} is EXP-complete under $\mathsf{P}_{/\mathrm{poly}}$-truth-table reductions and NP-Turing reductions [ABK+06], and these results use the non-relativizing "instance checkers" for EXP-complete problems [BFL91, BFNW93]. An *algebrization* barrier would be more satisfying for showing limitations of such techniques. However, we could not extend our oracle world to an algebrizing one in the sense of either [AW09], [IKK09], or [AB18].

Nevertheless, we managed to construct an oracle world where $\widetilde{\mathrm{Kt}}$ is computable in polynomial time, and $\mathsf{EXP} = \mathsf{ZPP}$ holds simultaneously.

Theorem 1.15. *There is a relativized world where* $\widetilde{\mathrm{Kt}}$ *complexity is computable in deterministic polynomial time, and* $\mathsf{EXP} = \mathsf{ZPP}$*.*

In this world, EXP-complete problems have trivial instance checkers, since they are in ZPP. We also get some other non-relativizing theorems such as IP = PSPACE for free, since PSPACE $\subseteq$ EXP = ZPP $\subseteq$ IP. As a result, we cannot prove that $\widetilde{\mathrm{Kt}}$ is not in polynomial time, even if we combine IP = PSPACE or the instance checkers for EXP-complete problems with relativizing techniques. We believe that this oracle world serves as a "fundamental obstacle" ([ABK+06]) to proving MKtP $\notin$ P.

We think our new complexity measure $\widetilde{\mathrm{Kt}}$ is of independent interest. Understanding $\widetilde{\mathrm{Kt}}$ using nonrelativizing techniques may serve as the first step towards solving Question 1.5.

1.2.3 Natural Proofs Versus Cryptography

Our third set of results is motivated by Question 1.11. Under the so-called "Universality Conjecture", [San20] answered Question 1.11 affirmatively, i.e. the non-existence of natural proofs is equivalent to the existence of one-way functions. In contrast, we show that the answer of Question 1.11 is false in some relativized world, establishing a barrier for constructing one-way functions from nonexistence of natural proofs. We can even rule out *auxiliary-input* one-way functions (a primitive weaker than one-way functions) in our world.

Consequently, the Universality Conjecture fails in this world. As we will discuss in Section 1.3.3, in this world, the Universality Conjecture actually fails in a *very intuitive way*.

Theorem 1.16 (informal version)**.** *There is a relativized world where* $\mathsf{P}_{/\mathrm{poly}}$*-natural properties useful against* $\mathsf{SIZE}[2^{\delta n}]$ *do not exist, and auxiliary-input one-way functions do not exist either.*

The non-existence of natural proofs corresponds to the zero-error average-case hardness of MCSP [HS17]. We also extend our results by showing a relativized world where MCSP or MINKT is hard even for two-sided error heuristics.

Theorem 1.17 (informal version)**.** *There is a relativized world where* GapMCSP *is hard on average under some samplable distribution, and auxiliary-input one-way functions do not exist.*

Theorem 1.18 (informal version)**.** *There is a relativized world where* GapMINKT *is hard on average under some samplable distribution, and auxiliary-input one-way functions do not exist.*

Besides Question 1.11, we also show the following consequences based on our relativized worlds:

(Question 1.9) Extending the results in [Hir18] to the bounded-error case requires nonrelativizing techniques, if the underlying distribution for MINKT is still the uniform distribution. (This is because [LP20] showed the equivalence between the existence of one-way functions and the bounded-error average-case hardness of MINKT under the uniform distribution.)

(Question 1.12) It requires nonrelativizing techniques to show that GapMINKT $\in$ CZK, or even that GapMINKT can be solved on average by a CZK protocol, on infinitely many input lengths. This is because [OW93] showed that if auxiliary-input one-way functions do not exist, then CZK = BPP.

Note that the proof that if one-way functions exist then NP $\subseteq$ CZK [GMW91] is already nonrelativizing. On the other hand, we show that basing GapMINKT $\in$ CZK on the *non*existence of one-way functions also requires a nonrelativizing proof.

1.2.4 Limits of GapMINKT as an Oracle

We also present technical barriers for showing *stronger* reductions to the GapMINKT oracle, such as coNP-Turing reductions or $\mathsf{P}_{/\mathrm{poly}}$-Turing reductions.

We view (Turing) reductions to a promise problem $L = (L.\textsc{Yes}, L.\textsc{No})$ as machines that interact with an (adversarial) oracle, and tries to solve a problem L'. We say a reduction is

robust, if it works even if the adversary is *inconsistent* on queries not in the promise. That is, on queries outside $(L.\textsc{Yes} \cup L.\textsc{No})$, the adversary can sometimes return 0 and sometimes return 1. Furthermore, the adversary is allowed to see the input of L' or the nondeterministic branch the reduction is running on, and decide whether to return 0 or 1 accordingly.

We show that a reduction that is both robust and relativizing cannot solve Question 1.10 or (a harder version of) Question 1.4. However, as the requirement of robust reductions seem very strong, we mainly treat these results as *technical* barriers rather than *conceptual* barriers. It is also worth mentioning that we use the "Gap" in GapMINKT in a very crucial way.

Theorem 1.19 (informal version)**.** *Each of the following items* cannot *be proved by a reduction that is both robust and relativizing.*

- *Either* GapMINKT $\in \mathsf{coNP}$*, or* GapMINKT *is* NP*-complete under* coNP*-Turing reductions.*
- *Every problem in* DistNP *admits a polynomial-size two-sided error heuristic with* GapMINKT *oracles.*

We did not manage to prove non-hardness results under $\mathsf{coNP}_{/\mathrm{poly}}$-Turing reductions, as mentioned in Question 1.4. We leave it as an open problem.

Open Problem 1.20. *Is there a relativized world where* GapMINKT $\notin \mathsf{coNP}_{/\mathrm{poly}}$*, and* GapMINKT *is not* NP*-complete under robust* $\mathsf{coNP}_{/\mathrm{poly}}$*-Turing reductions?*

1.3 Technical Overview

1.3.1 Meta-Complexity Problems Are Not Robust in Relativized Worlds

We briefly discuss the proof techniques of the first bullet of Theorem 1.13 here, i.e. there is an oracle world such that MCSP is easy but search-MCSP is hard. The framework for the other two bullets will be similar.

Making MCSP **easy.** We can add an MCSP oracle in our oracle world, but the circuit minimization problem in our world becomes $\mathrm{MCSP}^{\mathrm{MCSP}}$. Then we also need to add an $\mathrm{MCSP}^{\mathrm{MCSP}}$ oracle, but again, the circuit minimization problem becomes $\mathrm{MCSP}^{\mathrm{MCSP}^{\mathrm{MCSP}}}$ now. Therefore, a natural approach is to add the "limit" of

$$\mathrm{MCSP}^{\mathrm{MCSP}^{\mathrm{MCSP}^{\cdots}}}$$

into our oracle world. Indeed, this is what we do: We add an oracle itrMCSP (which stands for "iterated MCSP") into our world, such that (roughly speaking)

$$\mathrm{itrMCSP}[k, x, s] = \underbrace{\mathrm{MCSP}^{\mathrm{MCSP}^{\mathrm{MCSP}^{\cdots}}}}_{\text{iterate } k \text{ times}}[x, s].$$

(Recall that $\mathrm{MCSP}^{\mathcal{O}}[x, s] = 1$ if and only if in the oracle world with oracle $\mathcal{O}$, the circuit complexity of the truth table x is at most s.)

In our world, MCSP is indeed easy. Actually, let x be a truth table of length 2^n, then the circuit complexity of x is at most s in our world if and only if $\mathrm{itrMCSP}[2^n, x, s] = 1$.

Making search-MCSP **hard.** We define an oracle $\mathcal{O}$ that diagonalizes against every polynomial-time Turing machine M, and define itrMCSP relative to $\mathcal{O}$. (That is, for example, $\mathrm{itrMCSP}[1, x, s] = \mathrm{MCSP}^{\mathcal{O}}[x, s]$ and $\mathrm{itrMCSP}[2, x, s] = \mathrm{MCSP}^{\mathrm{MCSP}^{\mathcal{O}}}[x, s]$.) For every Turing machine M, we find a large enough integer N and a hard truth table x_{hard} of length $\mathrm{poly}(N)$. Then we feed x_{hard} to M. How we answer the $\mathcal{O}$ queries of M is not important, but each time M makes a query

itrMCSP$[k, x, s]$, we *pretend x has the lowest possible circuit complexity*, and answer this query accordingly.

To be more precise, we fix the oracle $\mathcal{O}$ up to input length $N-1$ before we simulate M on input x_{hard}. This has the effect that for every integer k, truth table x, and parameter $s \leq N-1$, we already know whether itrMCSP$[k, x, s] = 1$ regardless of how we fix the rest of $\mathcal{O}$; see Claim 3.3. Then upon every query itrMCSP$[k, x, s]$, if $s \leq N-1$ we already know how to reply it; otherwise we simply reply 1.

At last, for every query itrMCSP$[k, x, s]$ where $s \geq N$ and we returned 1, we need to put the truth table x in the length-N slice of $\mathcal{O}$ so that its circuit complexity is indeed at most N. Since M only runs in polynomial time, and only probes very few positions of $\mathcal{O}$, we can indeed put it somewhere in $\mathcal{O}$ without letting M notice. We do not need to care about the parameter k here, as MCSP$[x, N] = 1$ implies itrMCSP$[k, x, N] = 1$ for every k.[3] To diagonalize against M, we also put x_{hard} into the length-N slice of $\mathcal{O}$, but in a place that M did not probe at all. In this way, we can guarantee that there is a size-N circuit for x_{hard}, but M fails to find it.

1.3.2 Barriers for Proving Hardness of Levin's Complexity

We first define the complexity $\widetilde{\mathrm{Kt}}$. For a string x, let $\widetilde{\mathrm{Kt}}(x)$ denote the minimum possible value of $|M| + \lfloor \log t \rfloor$, where after we run the machine M on the empty input for t steps, the content of some tape of M is exactly x. The difference between Kt and $\widetilde{\mathrm{Kt}}$ is that in the definition of Kt, we require M to halt after outputting x; while in the definition of $\widetilde{\mathrm{Kt}}$, x can be an intermediate step of the computation.

A fixed-point oracle. Our approach will be to find a "fixed-point" of $\widetilde{\mathrm{Kt}}$: an oracle $\mathcal{O}$ such that $\mathcal{O}[x] = \widetilde{\mathrm{Kt}}^{\mathcal{O}}(x)$ for every string x. Then, in the world with oracle $\mathcal{O}$, we can compute $\widetilde{\mathrm{Kt}}(x)$ by simply calling $\mathcal{O}[x]$.

We proceed in stages, and in stage n, we fix the strings that have $\widetilde{\mathrm{Kt}}$ complexity exactly n. We enumerate every (M, t) such that $|M| + \lfloor \log t \rfloor = n$, and run M for t steps. For every intermediate tape content x, if $\mathcal{O}[x]$ is not fixed yet, then we fix $\mathcal{O}[x] = n$. A natural problem is: how to respond to the $\mathcal{O}$ queries made by M? The answer is surprisingly simple: for every query $\mathcal{O}[y]$ that M makes, we already have $\widetilde{\mathrm{Kt}}(y) \leq n$ by definition, so if $\mathcal{O}[y]$ is not fixed to a value smaller than n yet, then we can return $\mathcal{O}[y] = n$ confidently! It is not hard to show that the oracle $\mathcal{O}$ is indeed a "fixed-point" of $\widetilde{\mathrm{Kt}}$.

Achieving $\mathsf{EXP} = \mathsf{ZPP}$. It is also simple to achieve $\mathsf{EXP} = \mathsf{ZPP}$ in the above oracle. To simulate exponential time, we give the zero-error probabilistic polynomial-time machine a "cheat" oracle Cheat that embeds the truth tables of a certain EXP-complete problem. It is natural to choose the EXP-complete problem as

$$L = \{(M, t) : M \text{ on empty input outputs 1 in time } t\},$$

since we can construct $\mathcal{O}$ and obtain the truth tables of L at the same time. We can reply arbitrarily when M queries the Cheat oracle.

Now we have a "fixed-point" oracle $\mathcal{O}$ such that $\mathcal{O}[x] = \widetilde{\mathrm{Kt}}^{\mathcal{O},\mathsf{Cheat}}(x)$ for every x. We also have a length-2^n truth table (of L), which we want to "embed" into Cheat. We can simply embed it into the length-$3n$ (say) slice of Cheat, as there are still many empty slots not asked in the construction of $\mathcal{O}$. Actually, the number of empty slots is so large (around $2^{3n} - 2^n \mathrm{poly}(n)$) that we can embed it "everywhere we can". A ZPP algorithm can simply guess a pointer in the length-$3n$ slice of Cheat, and it will likely point to the truth table of L.

[3] This depends on the low-level definition of itrMCSP, but it is true for our Definition 3.2.

1.3.3 Natural Proofs Versus Cryptography

We only discuss how we prove Theorem 1.16. Our starting point is an oracle world in [Wee06, Section 5], in which there is a hard-on-average problem but no auxiliary-input one-way functions. Given a function $f : \{0,1\}^n \to \{0,1\}^n$, chosen uniformly at random, the world consists of two oracles: A PSPACE-complete oracle, and an "verification" oracle for f:

$$V_f[x,y] = \begin{cases} 1 & \text{if } f(x) = y, \\ 0 & \text{otherwise.} \end{cases}$$

Inverting auxiliary-input one-way functions. We use essentially the same argument as in [Wee06]. Roughly speaking, given any circuit C of size s, it is possible to "eliminate" every V_f gate in C, and obtain a circuit C' of size $\mathrm{poly}(s)$, such that C and C' agree on a $1 - 1/s$ fraction of inputs, but C' does not use V_f at all. This is because V_f behaves like an oracle that is both random and sparse. Therefore, for each V_f gate, we only need to store its answers to the inputs that appear frequently, and V_f is likely zero on other inputs.

Now, given any circuit C, we want to "invert" C, i.e. given $C(\mathbf{z})$ for a uniformly random input $\mathbf{z}$, output any string in $C^{-1}(C(\mathbf{z}))$. We simply find a circuit C' that is close to C, uses no V_f gates, and is only polynomially larger than C. Then we use the PSPACE-complete oracle to invert C'.

Ruling out natural proofs. It suffices to show there is a *succinct pseudorandom distribution*, i.e. a distribution $\mathcal{D}$ over truth tables with small circuits, such that $\mathcal{D}$ is indistinguishable from the uniform distribution by small circuits. (Actually, this approach is inspired by recent circuit lower bounds [HS17, CKLM20] for MCSP.)

Let $\mathcal{D}$ be any distribution over $\mathrm{poly}(s)$ strings, that fools PSPACE-oracle circuits of size s. The existence of $\mathcal{D}$ can be proven by the probabilistic method. For each $x \in \{0,1\}^{O(\log s)}$, let D_x be the x-th truth table in $\mathcal{D}$. We "embed" D_x into the oracle $V_f[x, f(x)]$, as follows:

$$V_f[x,y,\beta] = \begin{cases} D_x[\beta] & \text{if } f(x) = y, \\ \perp & \text{otherwise.} \end{cases}$$

Here, $D_x[\beta]$ is the β-th bit of D_x. Now we have artificially made $\mathcal{D}$ a *succinct* distribution: the circuit complexity of every string in $\mathcal{D}$ is small. We also need to prove $\mathcal{D}$ is *pseudorandom*, i.e. it fools every size $s^{o(1)}$ circuit. For every circuit C with V_f gates and PSPACE gates, we use the same method as above to eliminate every V_f gate in C, to obtain a circuit C' that is close to C. Note that the distribution under which we measure the closeness of C and C' is a hybrid of $\mathcal{D}$ and the uniform distribution. After that, we can use the fact that $\mathcal{D}$ fools C' to also show that $\mathcal{D}$ fools C, therefore C cannot be a natural proof.

How did the Universality Conjecture fail? The Universality Conjecture of [San20] roughly says that if there are succinct pseudorandom distributions, then there are *efficiently samplable* succinct pseudorandom distributions. However, in our oracle world, the succinct pseudorandom distribution $\mathcal{D}$ does not appear to be efficiently samplable: to sample from $\mathcal{D}$, it seems that we need be able to compute f, which is hard when f is a random function.

1.3.4 Limits of GapMINKT as an Oracle

At the core of our proofs is the following weakness of GapMINKT: *It may hide a small change of the oracle.* In particular, suppose we have two oracles $\mathcal{O}$ and $\mathcal{O}'$, such that they only differ at one input, then the "Gap" in GapMINKT allows us to choose an instantiation of GapMINKT that is both consistent with $\mathrm{GapMINKT}^{\mathcal{O}}$ and $\mathrm{GapMINKT}^{\mathcal{O}'}$. (See Lemma 6.2.) This instantiation

of GapMINKT would not help the reduction distinguish between $\mathcal{O}$ and $\mathcal{O}'$ at all; however, an NP problem on $\mathcal{O}$ and $\mathcal{O}'$ may have very different answers.

NP-intermediateness under coNP-Turing reductions. It is not hard to construct a relativized world where GapMINKT $\notin$ coNP (see, e.g. [Ko91, Theorem 4.1]). For the "non-completeness" part, we construct a diagonalizing oracle $\mathcal{O}$ such that there is no robust reduction from the NP problem

$$L = \{0^n : \mathcal{O} \cap \{0,1\}^n \neq \varnothing\}$$

to GapMINKT. On input length N, we construct a GapMINKT oracle that is both consistent with "$\mathcal{O} \cap \{0,1\}^N = \varnothing$" and "$|\mathcal{O} \cap \{0,1\}^N| = 1$". This oracle does not reveal whether $0^N \in L$, and we can still use the standard method to diagonalize against every co-nondeterministic Turing machine. In particular, we run this machine and reply 0 to all its queries to $\mathcal{O}$. If it rejects some branch, we put a string of length N that is not probed in this branch into $\mathcal{O}$; otherwise we do nothing.

Non-DistNP-hardness under $\mathsf{P}_{/\mathrm{poly}}$-Turing reductions. [GGKT05] showed that a random permutation $\pi : \{0,1\}^n \to \{0,1\}^n$ cannot be computed on average by circuits of size $2^{o(n)}$, even with a verification oracle

$$\Pi[\alpha, \beta] = \begin{cases} 1 & \text{if } \pi(\alpha) = \beta, \\ 0 & \text{otherwise.} \end{cases}$$

We show the same thing for (robust) circuits with Π and GapMINKT oracle gates. To over-simplify, the argument boils down to the following task: Given an input α, a circuit C that computes π correctly on α, and every value $\{\pi(\beta)\}_{\beta \neq \alpha}$, recover $\pi(\alpha)$. Without GapMINKT gates, it suffices to use $\log |C|$ bits to store a number k, such that on input α, the k-th Π gate of C contains the correct answer $\pi(\alpha)$. (For comparison, the trivial solution needs to record $n \gg \log |C|$ bits.)

Now, the circuit C has GapMINKT gates, and it is robust in the sense that $C^{\Pi,B}(\alpha) = \pi(\alpha)$ for *every* oracle B consistent with GapMINKT. Now we let B' be the MINKT oracle in the world where $\Pi[\alpha, \pi(\alpha)] = 0$, and other entries of Π are not changed. As the new oracle Π does not depend on $\pi(\alpha)$ at all, we can simulate $C^{\Pi,B'}(\alpha)$ without knowing $\pi(\alpha)$. On the other hand, we only modified one entry in Π, therefore B' is still consistent with GapMINKT. We still record the number k defined above for the simulation $C^{\Pi,B'}(\alpha)$, which suffices to recover $\pi(\alpha)$.

1.4 Related Works

In the paper that defined MINKT, Ko [Ko91] studied the properties of MINKT in relativized worlds. Among other results, [Ko91] showed that there is a relativized world where MINKT is neither in coNP, nor NP-complete under polynomial-time Turing reductions. This result indicates that the MINKT counterpart of Question 1.3 cannot be shown affirmatively using relativizing techniques. Also, [Ko91] constructed a relativized world where NP $\neq$ coNP, but MINKT *is* NP-complete under coNP-Turing reductions ("$\leq_T^{\mathsf{SNP}}$-reductions"). This leads to the conjecture [Ko91, Hir18] that MINKT might be NP-complete under coNP-Turing reductions in the unrelativized world (Question 1.4).

Our results in Section 5 build upon the results of Wee [Wee06]. The motivation of [Wee06] was to show that a certain cryptographic object (succinct noninteractive argument, SNARG) does not imply one-way functions in a relativizing way. The framework of [Wee06] was very helpful for us, as we also need to rule out (auxiliary-input) one-way functions. Moreover, it turns out that the SNARG constructed in [Wee06, Section 4] can be transformed into hardness of MCSP (Section 5.3)!

We also mention the negative results of Hirahara and Watanabe [HW16] that has a different but similar setting compared to ours. In particular, they consider reductions to MCSP (in the unrelativized world) that are *oracle-independent*, i.e. work for MCSP^A for every oracle A. Two particular results in [HW16] are that deterministic oracle-independent reductions cannot reduce problems outside P to MCSP, and that randomized oracle-independent reductions that only make one query cannot reduce problems outside $\mathsf{AM} \cap \mathsf{coAM}$ to MCSP. As discussed in [HW16], the difference between relativization and their model is that in the relativized world with A oracle, a Turing reduction has access to not only MCSP^A but also A itself; however in their model, the reduction does not have access to A.

2 Preliminaries

In this paper, logarithms are base 2 by default. We use boldface letters such as $\mathbf{x}, \mathbf{y}, \mathbf{z}$ to denote random variables. Let $\mathcal{D}$ be a distribution, then $\mathbf{x} \leftarrow \mathcal{D}$ means that $\mathbf{x}$ is a random variable drawn from the distribution $\mathcal{D}$. $\mathcal{U}_n$ denotes the uniform distribution over $\{0,1\}^n$.

We assume familiarity with the basic complexity classes, including NP, coNP, $\mathsf{P}_{/\text{poly}}$, ZPP, EXP, etc. A good reference for these complexity classes can be found in [AB09, Gol08]. A *distributional* problem $(L, \mathcal{D})$ is a pair of a problem L and a distribution ensemble $\{\mathcal{D}_n\}_{n\in\mathbb{N}}$, where each $\mathcal{D}_n$ is a distribution over the length-n inputs. DistNP is the class of distributional problems $(L, \mathcal{D})$ where $L \in \mathsf{NP}$ and $\mathcal{D}$ is efficiently samplable. (That is, there is a randomized algorithm that given input 1^n, outputs a string distributed according to $\mathcal{D}_n$.) AvgP is the class of distributional problems $(L, \mathcal{D})$ that admits polynomial-time errorless heuristics. (That is, there is a polynomial-time algorithm that on input $(x, 1^{\delta^{-1}})$, outputs either $L(x)$ or "don't know", such that the probability that it outputs "don't know" on a random input $\mathbf{x} \leftarrow \mathcal{D}_n$ is at most δ.)

In oracle worlds, the size of a circuit is defined as the number of wires in this circuit [Wil85]. There can be NOT gates on each wire of the circuit, and we do not count them into circuit complexity.

Let $f : \{0,1\}^n \to \{0,1\}$ be a function, we denote $\mathsf{CC}(f)$ as the circuit complexity of f, i.e. the size of the smallest circuit computing f. For oracles $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_k$, let $\mathsf{CC}^{\mathcal{O}_1,\mathcal{O}_2,\ldots,\mathcal{O}_k}(f)$ denote the circuit complexity of f, where the circuit has access to $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_k$ oracle gates of any fanin. We do not distinguish between a function and its truth table, so given a truth table x of length 2^n, $\mathsf{CC}(x)$ is the circuit complexity of (the function with truth table) x, and $\mathsf{CC}^{\mathcal{O}_1,\mathcal{O}_2,\ldots,\mathcal{O}_k}(x)$ is the circuit complexity of x with oracles $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_k$.

Let $\mathcal{O} : \{0,1\}^\star \to \{0,1\}$ be an oracle, we say the *length-n slice* of $\mathcal{O}$ is the function $\mathcal{O} : \{0,1\}^n \to \{0,1\}$, i.e. $\mathcal{O}$ restricted on length-n inputs.

We make the following assumptions in our model of oracle Turing machines. Suppose there is an oracle Turing machine with time complexity t and space complexity s, then:

- the total length of queries made to the oracles is at most t, and
- every query that the machine makes are of length at most s.

Note that this assumption is natural only when s is greater than the input length, which will always be the case in this paper.[4]

2.1 Meta-Computational Problems

The meta-computational problems we deal with in this paper are the Minimum Circuit Size Problem (MCSP, [KC00]), the Minimum Kt Problem (MKtP, [Lev84, ABK+06]), and the min-

[4] When $s = o(n)$, it is sometimes problematic to assume that an oracle Turing machine with space complexity s can only make queries of length at most s. For example, consider a logspace Turing reduction that shows a certain problem L is P-complete. On input length n, this reduction needs to make oracle queries to L with length $n^{\Omega(1)}$.

imum time-bounded Kolmogorov complexity problem (MINKT, [Ko91]).

2.1.1 Minimum Circuit Size Problem

Definition 2.1 (Minimum Circuit Size Problem)**.** The Minimum Circuit Size Problem (MCSP) is defined as follows:

- *Input:* a truth table x of length 2^n (that corresponds to a function over n input bits), and a size parameter s.
- *Output:* 1 if $\mathsf{CC}(x) \leq s$, and 0 otherwise.

We will also consider the parameterized version of MCSP.

Definition 2.2 (Parameterized Minimum Circuit Size Problem)**.** Let $s : \mathbb{N} \to \mathbb{R}$ be a size function, the problem MCSP$[s(n)]$ is defined as follows:

- *Input:* a truth table x of length 2^n.
- *Output:* 1 if $\mathsf{CC}(x) \leq s(n)$, and 0 otherwise.

For MCSP, we maintain the convention that N is the length of the input truth table, n is the arity of input function $f : \{0,1\}^n \to \{0,1\}$, and $N = 2^n$.

2.1.2 Minimum Kt Problem

Definition 2.3 (Levin's Kt Complexity)**.** Let x be a binary string, we define $\mathrm{Kt}(x)$ to be the smallest value of $|M| + \lfloor \log t \rfloor$, over all machines M and integers t such that on the empty input, M outputs x in t steps.

Definition 2.4 (Minimum Kt Complexity Problem)**.** The Minimum Kt Problem (MKtP) is defined as follows:

- *Input:* a binary string x and an integer k.
- *Output:* 1 if $\mathrm{Kt}(x) \leq k$, and 0 otherwise.

2.1.3 Minimum Time-Bounded Kolmogorov Complexity Problem

Definition 2.5 (Time-Bounded Kolmogorov Complexity)**.** Let x be a binary string and $t \geq |x|$ be an integer. We define the *t-time-bounded Kolmogorov complexity* of x, denoted as $\mathrm{K}^t(x)$, to be the length of the shortest program that outputs x in time t.

Definition 2.6 (Minimum Time-Bounded Kolmogorov Complexity Problem)**.** The Minimum Time-Bounded Kolmogorov Complexity Problem (MINKT) is defined as follows:

- *Input:* a binary string x, and two integers s and t in unary. (That is, $(x, 1^s, 1^t)$.)
- *Output:* 1 if $\mathrm{K}^t(x) \leq s$, and 0 otherwise.

2.1.4 Natural Proofs and Average-Case Complexity of MCSP

We also define natural proofs [RR97], which correspond to zero-error average-case heuristics for MCSP [HS17] under the uniform distribution. We will use the terms "natural proof" and "natural property" interchangeably.

Definition 2.7 (Natural Proofs)**.** Let $\mathcal{D}$ be a complexity class (such as P or $\mathsf{P}_{/\mathrm{poly}}$), $0 < \delta(N) < 1$ be a density parameter, and $s(n)$ be a size parameter. A *$\mathcal{D}$-natural property* with density $\delta(N)$ that is useful against $\mathsf{SIZE}[s(n)]$ is an algorithm $\mathcal{P}$, whose inputs are truth tables of size $N = 2^n$, such that:

(Constructivity) $\mathcal{P}$ can be implemented in complexity class $\mathcal{D}$;

(Largeness) $\mathcal{P}$ accepts at least a $\delta(N)$ fraction of length-N truth tables; and

(Usefulness) $\mathcal{P}$ rejects every truth table of length N that has circuit complexity at most $s(n)$.

For example, a P-natural property $\mathcal{P}$ useful against $\mathsf{SIZE}[2^{\epsilon n}]$ is simply a *one-sided error heuristic* for $\mathrm{MCSP}[2^{\epsilon n}]$ under the uniform distribution: $\mathcal{P}$ is allowed to err on a $1 - \delta(N)$ fraction of inputs by recognizing a hard truth table as "easy", but it is not allowed to recognize any easy truth table as "hard".

Since the fraction of truth tables that are easy is very small, [HS17] observed that natural proofs also correspond to *zero-error* average-case heuristics for MCSP, under the uniform distribution. For example, a P-natural property $\mathcal{P}$ useful against $\mathsf{SIZE}[2^{\epsilon n}]$ can be seen as a zero-error heuristic for $\mathrm{MCSP}[2^{\epsilon n}]$, that outputs "hard" on the truth tables it accepts, and *outputs "don't know"* on the truth tables it rejects. The converse is also true: a zero-error heuristic $\mathcal{A}$ for $\mathrm{MCSP}[2^{\epsilon n}]$ can be transformed into a P-natural property, by accepting every truth table that $\mathcal{A}$ outputs "hard", and rejecting every truth table that $\mathcal{A}$ outputs either "easy" or "don't know".

In this paper, when we discuss about zero-error average-case complexity for MCSP, unless stated otherwise, we will assume the underlying distribution is the uniform distribution. As argued in [HS17], it is very natural to characterize the zero-error average-case complexity of parameterized MCSP (i.e. $\mathrm{MCSP}[s(n)]$ for a function $s(\cdot)$) under the uniform distribution.

2.2 Projections and Single-Gate Circuits

A *projection* is a (multi-output) function where every output bit depends on at most one input bit. For example, if the projection receives n input bits, denoted as $x_1, x_2, \ldots, x_n$, then every output bit of the projection is in

$$\{0, 1\} \cup \{x_1, x_2, \ldots, x_n\} \cup \{\neg x_1, \neg x_2, \ldots, \neg x_n\}. \tag{1}$$

In Section 3, We will frequently consider *single-gate circuits*, i.e. oracle circuits with only one oracle gate. Let n be the number of input bits, s be the circuit complexity, $\mathcal{O}$ be the length-s slice of the oracle, then we can specify a single-gate circuit C by a projection $\mathsf{proj} : \{0,1\}^n \to \{0,1\}^s$, and a bit $b \in \{0, 1\}$. More precisely, for every $x \in \{0,1\}^n$,

$$C(x) = \mathcal{O}(\mathsf{proj}(x)) \oplus b.$$

We use the following fact regarding the number of single-gate circuits:

Fact 2.8. *Let $\mathcal{O}$ be an oracle, and n, s be integers. The number of different n-input size-s single-$\mathcal{O}$-gate circuits is exactly $(2n + 2)^s \cdot 2$.*

Proof Sketch. By Eq. (1), there are $2n + 2$ choices for each output bit of proj; there are two choices for b. □

2.3 Auxiliary-Input One-Way Functions

In this paper, we only consider the weakest definition of auxiliary-input one-way functions (AIOWFs) in [OW93, Vad06], where for every adversary $\mathcal{A}$, there are infinitely many auxiliary-inputs I (possibly depending on $\mathcal{A}$) on which $\mathcal{A}$ fails to invert. We will rule out AIOWFs in Section 5, so considering the weakest definition makes our results stronger.

Definition 2.9. An *auxiliary-input one-way function* is a collection of functions $\{f_x : \{0,1\}^{p(|x|)} \to \{0,1\}^{q(|x|)}\}_{x\in\{0,1\}^\star}$, where $p(\cdot), q(\cdot)$ are polynomials, such that

(Easy to compute) There is a polynomial-time algorithm F that given inputs $x \in \{0,1\}^\star$, $y \in \{0,1\}^{p(|x|)}$, outputs $f_x(y)$.

(Hard to invert) For every polynomial-time adversary $\mathcal{A}$, there is a negligible function μ (that is, $\mu(n) < \frac{1}{n^{\omega(1)}}$), and an infinite set $I \subseteq \{0,1\}^\star$, such that for every $x \in I$,

$$\Pr_{\mathbf{y}\leftarrow\mathcal{U}_{p(|x|)}}[\mathcal{A}(x, f_x(\mathbf{y})) \in f_x^{-1}(f_x(\mathbf{y}))] < \mu(|x|).$$

In Section 5, we will construct relativized worlds without AIOWFs. To prove there are no AIOWFs, it suffices to present a polynomial-time algorithm $\mathcal{A}$ that inverts every circuit.

Claim 2.10. *Suppose there is a polynomial-time algorithm $\mathcal{A}$ such that for every circuit $C : \{0,1\}^p \to \{0,1\}^q$ of size at most s,*

$$\Pr_{\mathbf{y}\leftarrow\mathcal{U}_p}[\mathcal{A}(C, 1^s, C(\mathbf{y})) \in C^{-1}(C(\mathbf{y}))] \geq 1 - O(1/s).$$

Then there are no auxiliary-input one-way functions. The proof relativizes.

Proof Sketch. Consider a candidate auxiliary-input one-way function $\{f_x\}_{x\in\{0,1\}^\star}$, the following adversary $\mathcal{A}_f$ inverts $\{f_x\}$ on almost every string x: On input $x, f_x(\mathbf{y})$, it constructs a polynomial-size circuit C that given y, computes $f_x(y)$, and then outputs $\mathcal{A}(C, 1^{|C|}, f_x(\mathbf{y}))$. □

2.4 Useful Lemmas

Existence of hard strings. In our constructions we will frequently use hard truth tables. A basic fact is that there are some (in fact, a lot of) truth tables that have large complexity.

Claim 2.11. *Let k be a constant, $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_k$ be oracles. For every large enough integer s, let $n = \lceil \log(3.1 s \log s) \rceil$, then there is a function $f : \{0,1\}^n \to \{0,1\}$ such that $\mathsf{CC}^{\mathcal{O}_1,\mathcal{O}_2,\ldots,\mathcal{O}_k}(f) > s$.*

Proof. We show that every circuit of wire complexity s can be encoded in $(3\log s + 1 + \log(k+2))s < 3.1 s \log s$ bits. For every gate, we use $\log(k+2)$ bits to write its type (AND, OR, $\mathcal{O}_1$, $\ldots$, $\mathcal{O}_k$). For every wire w, we use $3\log s + 1$ bits to write down four integers f, t, i, b, where w is an output wire of f and the i-th input wire of t, and b is a bit indicating whether there is a NOT gate on w. Therefore, there are less than $2^{3.1 s \log s}$ circuits with complexity s. Since there are $2^{2^n} \geq 2^{3.1 s \log s}$ functions on n inputs, the claim is true. □

Claim 2.12. *Let $\mathcal{O}$ be any oracle, s, t be two integers. Then there is a string x of length $s+1$ such that $\mathrm{K}^{t,\mathcal{O}}(x) > s$.*

Proof Sketch. The number of strings x such that $\mathrm{K}^{t,\mathcal{O}}(x) \leq s$ is at most $\sum_{i=0}^{s} 2^i = 2^{s+1} - 1$. □

Kolmogorov-random infinite strings. Let x be a (finite) binary string, we let $\mathrm{C}(x)$ be the length of the shortest program that outputs x on the empty input. (As opposed to the definition of time-bounded Kolmogorov complexity or Kt complexity, we do not place any requirement on the time needed to execute this program except that it is finite.) We also denote $\mathrm{C}(x \mid y)$ as the length of the shortest program that outputs x, given the information y. We need the existence of Kolmogorov-random infinite strings:

Theorem 2.13 ([LV08, Theorem 2.5.4])**.** *There is an infinite binary string ω such that for almost every integer n,*

$$\mathrm{C}(\omega_{1\sim n} \mid n) \geq n - 2\log n,$$

where $\omega_{1\sim n}$ is the first n bits of ω.

These random infinite strings will be used to construct "probabilistic" oracle worlds. Suppose we proved that some properties hold with high probability if our oracles are chosen in some random way. Then, to rigorously instantiate one such oracle world, we use a Kolmogorov-random infinite string ω as the "randomness" in our construction. We use the *incompressibility method* to show that these properties continue to hold in our world based on ω: otherwise, we can select an integer N and compress $\omega_{1\sim N}$ in less than $N - 2\log N$ bits.

Goldreich-Levin theorem. Let x, y be two strings of length n. We denote their *inner product* as $\langle x, y\rangle$, which is defined as $\langle x, y\rangle = \sum_{i=1}^{n} x_i \cdot y_i \bmod 2$. We will use the famous Goldreich-Levin theorem:

Theorem 2.14 ([GL89])**.** *Let $\epsilon > 0$ be a parameter. For every string $x \in \{0,1\}^n$, let $\mathcal{O}_x$ be an oracle such that $\mathcal{O}_x[r] = \langle x, r\rangle$. There is an algorithm A such that, for every hidden input x and every oracle $\widetilde{\mathcal{O}}$ that $(1/2+\epsilon)$-approximates $\mathcal{O}_x$, with probability at least $2/3$, $A^{\widetilde{\mathcal{O}}}(1^n, 1^{\lceil 1/\epsilon\rceil})$ outputs a list of $O(n/\epsilon^2)$ strings one of which is equal to x. Moreover, A runs in time $O(n^3/\epsilon^4)$.*

We will actually use the following corollary, that constructs a hard *Boolean* function from a hard-to-compute permutation:

Corollary 2.15. *Let $\epsilon > 0$ be a parameter, $\pi : \{0,1\}^n \to \{0,1\}^n$ be a permutation, then the following relativizes. Suppose that given $\alpha, \beta \in \{0,1\}^n$, we can check whether $\pi(\alpha) = \beta$ in $O(n)$ size. Also suppose that no circuit of size s computes π successfully on an ϵ fraction of inputs. Then the following problem L^{hard} is $(1/2+2\epsilon)$-hard for circuits of size $\frac{s}{(n/\epsilon)^{10}}$.*

$$L^{\mathsf{hard}} = \{(\alpha, r) : \langle \pi(\alpha), r\rangle = 1\}.$$

Hoeffding bound. We also need the following bound:

Theorem 2.16 ([Hoe63])**.** *Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ be independent random variables such that for every $1 \le i \le n$, we have $a_i \le \mathbf{x}_i \le b_i$. Let $\mathbf{X} = \sum_{i=1}^{n} \mathbf{x}_i$, then for every $t > 0$, we have*

$$\Pr[\mathbf{X} - \mathbb{E}[\mathbf{X}] \ge t] \le e^{-2t^2/\sum_{i=1}^{n}(b_i - a_i)^2}.$$

Miscellaneous. The following inequality will be useful.

Claim 2.17. *Let k, n be integers such that $0 \le k < n/2$, then $\log \binom{n}{k} \le k(2 - \log(k/n))$.*

Proof Sketch. Let $H(\epsilon) = -(\epsilon \log \epsilon + (1-\epsilon)\log(1-\epsilon))$ be the binary entropy function. The claim is immediate from the following inequalities: $\log \binom{n}{k} \le H(k/n)n$ and $H(\epsilon) < \epsilon \log(4/\epsilon)$. □

3 Relativized Worlds Separating Variants of MCSP

In this section, we show relativized worlds that support the following claim: A *slight change* in the definition of a meta-complexity problem often results in a *completely different* problem. In particular, we will prove the following theorem, where in each sub-section of this section, we will prove one bullet of the theorem.

Theorem 3.1. *For each of the following items, there is a relativized world where it becomes true.*

- MCSP *is in linear time, but* search-MCSP *requires $2^{\Omega(N/\log N)}$ time to solve.*
- *Let $0 < \delta_1 < \delta_2 < 1$ be two constants such that $1/\delta_1 - 1/\delta_2 > 1$. MCSP$[2^{\delta_2 n}]$ is in linear time, but MCSP$[2^{\delta_1 n}]$ requires $2^{N^{\delta_1}}/N^{\omega(1)}$ time to solve.*
- *Let $0 < \delta_1 < \delta_2 < 1$ be two constants. MCSP$[2^{\delta_1 n}]$ admits a polynomial-time errorless heuristic, but any errorless heuristic for MCSP$[2^{\delta_2 n}]$ requires $2^{N^{\delta_2}}/N^{\omega(1)}$ time.*

3.1 search-MCSP vs. MCSP

In this section, we construct a relativized world where MCSP is computable in linear time, but search-MCSP requires $2^{\Omega(N/\log N)}$ time to solve.

Our world consists of two oracles: $\mathcal{O}$ and itrMCSP. Here, $\mathcal{O}$ is a carefully constructed oracle that diagonalizes against every $2^{0.01N/\log N}$-time Turing machine that attempts to solve search-MCSP, and itrMCSP is an "iterated MCSP" oracle defined as follows.

Definition 3.2. The oracle itrMCSP receives three inputs: a nonnegative integer k which is the "iteration number", a truth table x, and a size parameter s. Its output is 1 if and only if there is a circuit C of size at most s such that the following are true.

- The circuit C has oracle access to $\mathcal{O}$ and itrMCSP.
- The truth table of C is x.
- On every input to C, the iteration number (i.e. the first input parameter) of every call to itrMCSP is at most $k-1$. (In particular, if $k=0$, then we do not allow oracle access to itrMCSP.)

Given the oracle itrMCSP, it is easy to see that MCSP has a linear-time algorithm. (Recall that in this oracle world, $\mathrm{MCSP}[x,s]=1$ if and only if there is a circuit of size s *with access to oracles* $\mathcal{O}$ *and* itrMCSP, whose truth table is x.) Let x be a truth table of length N, and C be the optimal circuit for computing x. It is easy to see that $|C|\le N$, therefore whenever C makes any call to itrMCSP, the iteration number it feeds to itrMCSP is at most 2^N-1. It follows that $\mathrm{MCSP}[x,s]$ should return 1 if and only if $\mathrm{itrMCSP}[2^N,x,s]=1$. Therefore MCSP has a linear-time algorithm.

Now we need to show that search-MCSP is hard. Let $\{M_i\}_{i\in\mathbb{N}}$ be an enumeration of all Turing machines that run in $2^{0.01N/\log N}$ time. We proceed in stages, and diagonalize against one machine in each stage.[5] At the end of stage i, there will be an input length N_i such that every input to $\mathcal{O}$ with length at most N_i is *fixed*, i.e. we have decided whether $\mathcal{O}[x]=1$ for every string x of length at most N_i.

We can see that $\mathcal{O}$ and itrMCSP are well-define oracles: For every string x, let i be the first stage such that $N_i\ge|x|$, then $\mathcal{O}[x]$ is defined in stage i. Moreover, by Definition 3.2, itrMCSP is completely determined by $\mathcal{O}$.

Actually, we claim that after we fixed every input to $\mathcal{O}$ with length at most N_i, the portion of itrMCSP with circuit complexity at most N_i is already determined.

Claim 3.3. *Let $k\ge 0$ and $s\in[0,N_i]$ be integers, and x be a truth table. After stage i,* $\mathrm{itrMCSP}[k,x,s]$ *is already determined by the slices of $\mathcal{O}$ on input lengths at most N_i.*

Proof. We proceed by induction on k. We will use $k=-1$ as the base case of the induction, where the claim is vacuously true. Now suppose for every integer $0\le k'\le k-1$, every truth table x' and every size parameter $s'\le N_i$, we have already determined $\mathrm{itrMCSP}[k',x',s']$. Consider a truth table x and a size parameter $s\le N_i$.

Let C be any oracle circuit of size at most s, that has access to $\mathcal{O}$ and itrMCSP oracles. Suppose that for every invocation of $\mathrm{itrMCSP}[k',x',s']$ on every possible input of C, we have $0\le k'<k$; and the truth table of C is x. Since $s\le N_i$, every oracle gate in C has fanin at most N_i. Consider an oracle gate $g\in C$:

- Suppose g is an $\mathcal{O}$ gate. As the behavior of $\mathcal{O}$ on every input of length at most N_i is fixed, the behavior of g is fixed.

[5] We assume that each machine M appears in the list $\{M_i\}$ infinitely many times, therefore no such machine can solve search-MCSP on *all but finitely many* inputs. We also implicitly make this assumption in the subsequent sections.

- Suppose g is an itrMCSP gate. For every input (k', x', s') that g may receive, we have $k' < k$ and $|x'| \leq N_i$. If $s' \leq N_i$, then by induction hypothesis, itrMCSP$[k', x', s']$ is already fixed. If $s' > N_i$, then $s' > |x'|$, so itrMCSP$[k', x', s']$ is fixed to 1.

Therefore, the behavior of every circuit C of size at most N_i is fixed, as long as the iteration number k' of every possible invocation of itrMCSP by C is at most $k-1$. It follows that itrMCSP$[k, x, s]$ is also fixed. □

For each integer $i \geq 0$, we will diagonalize against M_i in stage $i+1$, as follows. Let $\widetilde{N}$ be the smallest power of 2 that is greater than $4N_i \log N_i$, and x_{hard} be a truth table of length $\widetilde{N}$ that is "complex enough". (We will discuss how to choose x_{hard} later; we guarantee that before stage $i+1$, the circuit complexity of x_{hard} is greater than N_i.) We run the machine M_i on input x_{hard}.

- Whenever M_i makes a query $\mathcal{O}[x]$, if $\mathcal{O}[x]$ is already fixed then we return $\mathcal{O}[x]$; otherwise we fix $\mathcal{O}[x]$ to be 0.
- Whenever M_i makes a query itrMCSP$[k, x, s]$, if $s \leq N_i$ then we return the already determined value itrMCSP$[k, x, s]$ by Claim 3.3; otherwise we return 1, indicating the circuit complexity of x is at most s.

Machine M_i runs in $2^{0.01\widetilde{N}/\log \widetilde{N}} < 2^{0.09N_i}$ time, and outputs an oracle circuit C that attempts to compute x_{hard}. We will set the oracle $\mathcal{O}$ such that the final circuit complexity of x_{hard} will be exactly $N_i + 1$, so we may assume that the size of C is exactly $N_i + 1$. We simulate the circuit C on every input in $\{0,1\}^{\log \widetilde{N}}$, and use exactly the same method as before to answer its queries to $\mathcal{O}$ or itrMCSP. (That is, every query $\mathcal{O}[x]$ not fixed so far is fixed to be 0, and every query itrMCSP$[k, x, s]$ such that $s > N_i$ is fixed to be 1.)

- Let $\mathcal{Q}_{\mathcal{O}}$ be the set of queries made by either M_i or C to $\mathcal{O}$, whose lengths are at least $N_i + 1$. These are the queries to $\mathcal{O}$ that we have already fixed to 0.
- Let $\mathcal{Q}_{\text{itrMCSP}}$ be the set of truth tables in each query made by either M_i or C to itrMCSP, whose circuit complexity parameter s is at least $N_i + 1$. We need to "embed" every truth table in $\mathcal{Q}_{\text{itrMCSP}}$ into $\mathcal{O}$ such that they can be computed by a single-$\mathcal{O}$-gate circuit with fanin $N_i + 1$. We also require that the circuit complexity of x_{hard} is exactly $N_i + 1$; we simply add x_{hard} into the set $\mathcal{Q}_{\text{itrMCSP}}$.

Every truth table in $\mathcal{Q}_{\text{itrMCSP}}$ has length at most $2^{0.09N_i}$. Now we "embed" them into $\mathcal{O}$ as follows. For every truth table $x \in \mathcal{Q}_{\text{itrMCSP}}$, we have $\mathsf{CC}^{\mathcal{O},\text{itrMCSP}}(x) \geq N_i + 1$, and we want that its circuit complexity is exactly $N_i + 1$. We assign a distinct "prefix" pfx_x of length $0.91N_i + 1$ to this query. For every $0 \leq j < |x|$, we set

$$\mathcal{O}[\mathsf{pfx}_x \circ j \circ 0^{N_i+1-|\mathsf{pfx}_x|-\log|x|}] = x_j, \forall j \in \{0,1\}^{\log|x|}. \tag{2}$$

Here, $\circ$ denotes string concatenation operator, and j is interpreted as both an integer in $[0, |x|)$ and a string of length $\log |x|$. We can see that the length of $\mathsf{pfx}_x \circ j \circ 0^{N_i+1-|\mathsf{pfx}_x|-\log|x|}$ is exactly $N_i + 1$.

Eq. (2) gives a circuit of size $N_i + 1$ (on input j) whose truth table is x, so the circuit complexity of x is indeed $N_i + 1$. Finally, note that any string of length $0.91N_i + 1$ that is not a prefix of any string in $\mathcal{Q}_{\mathcal{O}}$ can be a valid prefix pfx_x. As $|\mathcal{Q}_{\mathcal{O}}| \leq 2^{0.09N_i}$, there are $2^{0.91N_i+1} - |\mathcal{Q}_{\mathcal{O}}| \gg 2^{0.09N_i} \geq |\mathcal{Q}_{\text{itrMCSP}}|$ such prefixes, so we can indeed assign a distinct prefix to every truth table in $\mathcal{Q}_{\text{itrMCSP}}$.

We have fixed the answer of some inputs of $\mathcal{O}$, either in $\mathcal{Q}_{\mathcal{O}}$, or by Eq. (2). (And the length of every such input is greater than N_i.) We define N_{i+1} to be the maximum length of these

inputs. Finally, every input to $\mathcal{O}$ with length in $[N_i+1, N_{i+1}]$ that is not fixed yet are fixed arbitrarily. This completes the description of stage $i+1$.

Now we prove that M_i does not compute search-MCSP correctly on input x_{hard}, for a suitably chosen x_{hard}. We define two oracles $\mathcal{O}^\star$ and itrMCSP* that simulate our strategy of answering queries:

- For any input x, if $|x| \le N_i$ then define $\mathcal{O}^\star[x] = \mathcal{O}[x]$; otherwise define $\mathcal{O}^\star[x] = 0$.
- For any input (k, x, s), if $s \le N_i$ then define itrMCSP$^\star[k, x, s]$ = itrMCSP$[k, x, s]$; otherwise define itrMCSP$^\star[k, x, s] = 1$.

Let x_{hard} be any truth table of length $\widetilde{N} \ge 4N_i \log N_i$ such that $\mathsf{CC}^{\mathcal{O}^\star,\text{itrMCSP}^\star}(x_{\mathsf{hard}}) \ge N_i+2$. The existence of x_{hard} is guaranteed by Claim 2.11. It is easy to see that if we choose x_{hard} in this way, M_i indeed fails to solve search-MCSP on input x_{hard}. Actually, let C be the output of M_i on input x_{hard}. If $|C| \ne N_i+1$, then clearly M_i is incorrect. If $|C| = N_i+1$, then C does not distinguish between $(\mathcal{O}, \text{itrMCSP})$ and $(\mathcal{O}^\star, \text{itrMCSP}^\star)$. However, by the choice of x_{hard}, we have that $\mathsf{CC}^{\mathcal{O}^\star,\text{itrMCSP}^\star}(x_{\mathsf{hard}}) \ge N_i+2$, so x_{hard} cannot be the truth table of C, thus M_i is still incorrect.

The search version of GapMCSP **is hard for deterministic algorithms.** Actually, we can also show that relativizing techniques do not suffice to show *deterministic* search-to-decision reductions for MCSP, even if the search algorithm is allowed to only provide an approximately smallest circuit. Actually, we can make the inapproximability gap *very* large: $\omega(n)$ vs. $2^n/4n$. In contrast, such an approximate-search-to-decision reduction is already known in [CIKK16, Hir18, San20], if we allow randomized solutions.

Theorem 3.4. *For every constant $c \ge 2$, there are oracles $\mathcal{O}$, itrMCSP such that the following hold.*

- MCSP$^{\mathcal{O},\text{itrMCSP}}$ *can be solved in linear time with an* itrMCSP *oracle.*
- *For every Turing machine M that runs in N^c time, there is a truth table x of length $N = 2^n$ and circuit complexity $\mathsf{CC}^{\mathcal{O},\text{itrMCSP}}(x) \le 4cn$, such that M on input x does not output an $(\mathcal{O}, \text{itrMCSP})$-oracle circuit of size $2^n/4n$ that computes x.*

Proof Sketch. We adapt the above construction. In particular, the stage $i+1$ is as follows.

First, let $\widetilde{N} = 2^{\lfloor (0.3/c)N_i \rfloor}$, and x_{hard} be a truth table of length $\widetilde{N}$ such that $\mathsf{CC}^{\mathcal{O}^\star,\text{itrMCSP}^\star}(x_{\mathsf{hard}}) > \widetilde{N}/4 \log \widetilde{N}$. Again, by Claim 2.11, such a truth table x_{hard} exists. We feed x_{hard} to the machine M_i, and whenever it asks queries to $\mathcal{O}$ or itrMCSP, we reply with $\mathcal{O}^\star$ or itrMCSP* instead. Then M_i outputs a circuit C of size at most $\widetilde{N}/4 \log \widetilde{N}$ attempting to compute x_{hard}.

Let $T = \widetilde{N}^c$, then M_i makes at most T queries to the oracles ($\mathcal{O}$ and itrMCSP). As $c \ge 2$, the circuit C on all of its possible inputs also makes at most T queries to the oracles in total.

Now, we have $2T$ queries to $\mathcal{O}$ that are fixed to be 0, and $2T+1$ truth tables (including x_{hard}) queried to itrMCSP, each of length at most T, which we want to embed into the length-(N_i+1) slice of $\mathcal{O}$. Since $T \le 2^{0.3N_i}$, this is possible. Let $n = \log \widetilde{N}$, then we have $\mathsf{CC}^{\mathcal{O},\text{itrMCSP}}(x_{\mathsf{hard}}) \le 4cn$, but the size-$2^n/4n$ circuit outputted by M_i does not compute x_{hard}. □

3.2 MCSP$[2^{\delta_1 n}]$ vs. MCSP$[2^{\delta_2 n}]$

Let $0 < \delta_1 < \delta_2 < 1$ be two constants such that $1/\delta_1 - 1/\delta_2 > 1$. The main result in this section is an oracle relative to which MCSP$[2^{\delta_1 n}]$ requires $2^{N^{\delta_1}}/N^{\omega(1)}$ time to solve, but MCSP$[2^{\delta_2 n}]$ can be computed in linear time. Note that in any relativized world, MCSP$[2^{\delta_1 n}]$ can always be solved in $2^{\widetilde{O}(N^{\delta_1})}$ time by brute force, so our result is nearly tight.

As the same in Section 3.1, we provide two oracles $\mathcal{O}$ and itrMCSP. The first oracle $\mathcal{O}$ will be constructed by careful diagonalization, and the second oracle itrMCSP is an "iterated" version of MCSP$[2^{\delta_2 n}]$. More precisely:

Definition 3.5. The oracle itrMCSP receives two inputs: a nonnegative integer k which is the "iteration number", and a truth table x. Its output is 1 if and only if there is a circuit C of size at most $\lfloor |x|^{\delta_2} \rfloor$ such that the following are true.

- The circuit C has oracle access to $\mathcal{O}$ and itrMCSP.
- The truth table of C is x.
- On every input to C, the iteration number (i.e. the first input parameter) of every call to itrMCSP is at most $k-1$. (In particular, if $k=0$, then we do not allow oracle access to itrMCSP.)

Again, itrMCSP is completely determined by $\mathcal{O}$.

The same reasoning as in Section 3.1 shows that MCSP$[2^{\delta_2 n}]$ is in linear time. So our task is to carefully design the oracle $\mathcal{O}$ such that MCSP$[2^{\delta_1 n}]$ is hard.

Let $\{M_i\}_{i\in\mathbb{N}}$ be an enumeration of Turing machines running in $2^{N^{\delta_1}}/N^{\omega(1)}$ time. We proceed in stages, where at the end of stage i, there is a number N_i such that every input to $\mathcal{O}$ of length at most N_i is fixed. Initially we set N_0 to be a large enough constant, and (arbitrarily) fix every input of length at most N_0.

We diagonalize against M_i in stage $i+1$. Every input to $\mathcal{O}$ of length N_i is already fixed, and by the same reasoning as in Claim 3.3, for every integer k and every truth table x with $\lfloor |x|^{\delta_2} \rfloor \le N_i$, we can answer itrMCSP$[k,x]$ with certainty. Therefore, we will call the following queries "fixed": every query to $\mathcal{O}$ with length at most N_i, and every query (k,x) to itrMCSP where $\lfloor |x|^{\delta_2} \rfloor \le N_i$.

In contrast to Section 3.1, we will show how to choose the hard input x_{hard} at first, and then diagonalize against M_i.

Let N_{short} be the unique power of 2 such that $\lfloor (N_{\mathsf{short}})^{\delta_2} \rfloor = N_i+1$. If N_{short} does not exist or is not unique, we increase N_i until there is a power of 2 in the interval $[(N_i+1)^{1/\delta_2}, (N_i+2)^{1/\delta_2})$. (When we increase N_i, we fix the length-N_i slice of $\mathcal{O}$ arbitrarily.) Let N_{long} be the smallest power of 2 such that $\lfloor (N_{\mathsf{long}})^{\delta_1} \rfloor \ge N_i+1$. As $\delta_1 < \delta_2$ and N_i is large enough, we have that $N_{\mathsf{short}} < N_{\mathsf{long}}$.

Choosing the input x_{hard}. Roughly speaking, we will feed a truth table x_{hard} of length N_{long} to M_i, and the truth table in every "meaningful" query that M_i asks to itrMCSP has length N_{short}.

Let x_{hard} be a truth table of length N_{long} that satisfies the following two properties:

- If the only oracle gates we use are $\mathcal{O}$ gates of fanin at most N_i, and itrMCSP oracle gates of fanin at most N_i+1, then the circuit complexity of x_{hard} is at least N_i+2.
- x_{hard} is a function that depends on all $(\log N_{\mathsf{long}})$ input bits.

The proof of Claim 2.11 can be adapted to show that a $1-o(1)$ fraction of length-N_{long} truth tables satisfy the first bullet. Also, a $1-o(1)$ fraction of length-N_{long} truth tables satisfy the second bullet. Therefore there exists a suitable x_{hard}.

Our plan is to run M_i on the input x_{hard}, and trick M_i into incorrectly deciding whether the circuit complexity of x_{hard} is at most N_i+1. By the choice of x_{hard}, we can see that if the circuit complexity of x_{hard} is at most N_i+1, then its complexity is exactly N_i+1, and it is computed by a single-$\mathcal{O}$-gate circuit. That is, there is a projection $\mathsf{proj} : \{0,1\}^{\log N_{\mathsf{long}}} \to \{0,1\}^{N_i+1}$ and a bit b, such that for every $j \in \{0,1\}^{\log N_{\mathsf{long}}}$,

$$(x_{\mathsf{hard}})_j = \mathcal{O}[\mathsf{proj}(j)] \oplus b.$$

Recall that x_{hard} is the truth table of a function that depends on all input bits. This means that proj also has to depend on all input bits (of j). As proj is a projection, it is also an *injection*. In the rest of this sub-section, we will implicitly assume every projection we consider is an injection.

A noise function $\mathcal{O}_{\mathsf{noise}}$. Before running M_i on the input x_{hard}, we need to construct a "noise" function, which is a partial function $\mathcal{O}_{\mathsf{noise}}$ on length-(N_i+1) inputs. Denote $\mathcal{S}$ as the domain of $\mathcal{O}_{\mathsf{noise}}$, then every input in $\{0,1\}^{N_i+1}$ is in $\mathcal{S}$ with probability $p_{\mathsf{noise}} = 1/N_{\mathsf{short}}$ independently. If an input x is in $\mathcal{S}$, then the value of $\mathcal{O}_{\mathsf{noise}}[x]$ is either 0 or 1 with equal probability independently.

In the following lemmas, we will show that with high probability, $\mathcal{O}_{\mathsf{noise}}$ "kills" any single-gate circuit trying to compute the length-N_{long} truth table x_{hard}, but leaves plenty room for embedding truth tables of length N_{short}.

Lemma 3.6. *With probability $1-o(1)$, any oracle consistent with $\mathcal{O}_{\mathsf{noise}}$ cannot "compute" x_{hard} by a single-gate circuit. More precisely, for every projection $\mathsf{proj} : \{0,1\}^{\log N_{\mathsf{long}}} \to \{0,1\}^{N_i+1}$ and bit $b \in \{0,1\}$, there is some $j \in \{0,1\}^{\log N_{\mathsf{long}}}$ such that $\mathsf{proj}(j) \in \mathcal{S}$, but $(x_{\mathsf{hard}})_j \neq \mathcal{O}_{\mathsf{noise}}[\mathsf{proj}(j)] \oplus b$.*

Proof. For a single-gate circuit corresponding to the projection proj and the bit b, the probability that it remains "intact", i.e. the above j does not exist, is at most $(1-p_{\mathsf{noise}}/2)^{N_{\mathsf{long}}}$. By Fact 2.8, there are at most $2 \cdot (2\log N_{\mathsf{long}} + 2)^{N_i+1}$ such single-gate circuits. By a union bound, the probability that x_{hard} is computed by some single-gate circuit is at most

$$\begin{aligned}
& 2 \cdot (2\log N_{\mathsf{long}} + 2)^{N_i+1} \cdot (1 - p_{\mathsf{noise}}/2)^{N_{\mathsf{long}}} \\
\leq\ & 2 \cdot e^{O(N_i \log\log N_{\mathsf{long}})} \cdot (1/e)^{N_{\mathsf{long}}/2N_{\mathsf{short}}} \\
\leq\ & \exp(O(N_i \log\log N_{\mathsf{long}}) - \Omega(N_{\mathsf{long}}/N_{\mathsf{short}})) \\
\leq\ & \exp\Big(O(N_i \log\log N_i) - \Omega(N_i^{1/\delta_1 - 1/\delta_2})\Big) \\
=\ & o(1). \qquad \text{(recall } 1/\delta_1 - 1/\delta_2 > 1\text{)}
\end{aligned}$$

□

Let pfx be a string of length $N_i + 1 - \log N_{\mathsf{short}}$. For every string $j \in \{0,1\}^{\log N_{\mathsf{short}}}$, recall that $\mathsf{pfx} \circ j$ is the length-(N_i+1) string obtained by concatenating pfx and j. We say pfx is a *good* prefix, if every string of the form $\mathsf{pfx} \circ j$ is not in $\mathcal{S}$. (That is, we can safely "embed" any length-N_{short} truth table in the size-N_{short} subcube indexed by pfx.)

Lemma 3.7. *With probability $1-o(1)$, there are at least $\Omega(2^{N_i}/N_{\mathsf{short}})$ good prefixes pfx.*

Proof. Suppose N_{short} is large enough, then any fixed prefix pfx is good with probability

$$(1 - p_{\mathsf{noise}})^{N_{\mathsf{short}}} = (1 - 1/N_{\mathsf{short}})^{N_{\mathsf{short}}} > 0.36.$$

Since whether a prefix pfx is good is independent from all other prefixes, the lemma follows from Chernoff bound. □

At the beginning of stage $i+1$, after choosing the input $x_{\mathsf{hard}} \in \{0,1\}^{N_{\mathsf{long}}}$, we will choose $\mathcal{O}_{\mathsf{noise}}$ such that the conclusions of both Lemma 3.6 and 3.7 hold.

Completing stage $i+1$. We run M_i on input x_{hard}. The time complexity of M_i is at most

$$2^{(N_{\mathsf{long}})^{\delta_1}}/(N_{\mathsf{long}})^{\omega(1)} \leq 2^{N_i}/N_i^{\omega(1)}.$$

Recall that the following queries are already "fixed": every query to $\mathcal{O}$ with length at most N_i, and every query (k,x) to itrMCSP with $\lfloor |x|^{\delta_2} \rfloor \leq N_i$. Each time M_i asks a query q, if it is "fixed", then we return the fixed value. Otherwise, if q is asked to $\mathcal{O}$, and $q \in \mathcal{S}$ (which implies

$|q| = N_i + 1$), then we answer $\mathcal{O}_{\mathsf{noise}}[q]$. Otherwise, no matter which oracle q is asked to, we answer 1.

Machine M_i asked some queries (k, x) to itrMCSP that are not fixed. Let $\mathcal{Q}_{\mathsf{useful}}$ be the set of those truth tables x such that $|x| = N_{\mathsf{short}}$ (i.e. $\lfloor |x|^{\delta_2} \rfloor = N_i + 1$), and $\mathcal{Q}_{\mathsf{useless}}$ be the rest truth tables, i.e. those with $\lfloor |x|^{\delta_2} \rfloor \geq N_i + 2$. Recall that we answered 1 on every query $x \in \mathcal{Q}_{\mathsf{useful}} \cup \mathcal{Q}_{\mathsf{useless}}$, so we need to ensure that the circuit complexity of x is at most $\lfloor |x|^{\delta_2} \rfloor$.

Embedding useless truth tables. We first embed every truth table $x \in \mathcal{Q}_{\mathsf{useless}}$ into the length-$\lfloor |x|^{\delta_2} \rfloor$ slice of $\mathcal{O}$. These queries are "useless" as they will not influence the length (N_i+1) slice of $\mathcal{O}$, and thus will not influence the circuit complexity of x_{hard}. Now we process the truth tables $x \in \mathcal{Q}_{\mathsf{useless}}$ one by one. Let $n_x = \lfloor |x|^{\delta_2} \rfloor$. We assign a prefix pfx_x of length $n_x + 2 - \log |x|$ to x, and embed x into $\mathcal{O}$ as

$$\mathcal{O}[\mathsf{pfx}_x \circ j] = x_j, \forall j \in \{0,1\}^{\log |x|}.$$

Every prefix pfx_x of length $n_x + 2 - \log |x|$ is valid, unless

(a) pfx_x is the prefix of some string queried by M_i to the length-n_x slice of $\mathcal{O}$, or

(b) $\mathsf{pfx}_x = \mathsf{pfx}_{x'}$, where x' is a truth table in $\mathcal{Q}_{\mathsf{useless}}$ processed before x, and $n_{x'} = n_x$.

Let $T = 2^{N_i}/N_i^{\omega(1)}$ be the time complexity of M_i. Each prefix satisfying (a) or (b) corresponds to a query made by M_i which has length $\Omega(|x|)$, therefore the number of such prefixes is at most $O(T/|x|)$. We still have

$$2^{n_x+2-\log |x|} - O(T/|x|) \geq 2^{N_i}/|x| - 2^{N_i}/(N^{\omega(1)}|x|) \gg 1$$

valid prefixes, and we can pick an arbitrary one as pfx_x. We can see that the circuit complexity of every truth table $x \in \mathcal{Q}_{\mathsf{useless}}$ is at most $\lfloor |x|^{\delta_2} \rfloor$.

Embedding useful truth tables. Now we embed every truth table in $\mathcal{Q}_{\mathsf{useful}}$ into the length-$(N_i + 1)$ slice of $\mathcal{O}$. There are two cases depending on the output of M_i on input x_{hard}.

Case I: M_i outputs 0 on x_{hard}. In this case, to make M_i wrong, we want that the circuit complexity of x_{hard} is at most $N_i + 1$.

Let $\mathcal{Q}_{\mathcal{O}}$ be the set of queries asked by M_i to the length-$(N_i + 1)$ slice of $\mathcal{O}$. Then $|\mathcal{Q}_{\mathcal{O}}| \leq 2^{N_i}/N_i^{\omega(1)}$. Our oracle $\mathcal{O}$ needs to be consistent with our responses on $\mathcal{Q}_{\mathcal{O}}$.

We first embed x_{hard} into the length-$(N_i + 1)$ slice of $\mathcal{O}$. We pick any string $\mathsf{pfx}_{x_{\mathsf{hard}}}$ of length $N_i + 1 - \log N_{\mathsf{long}}$ such that $\mathsf{pfx}_{x_{\mathsf{hard}}}$ is not the prefix of any string in $\mathcal{Q}_{\mathcal{O}}$. As $|\mathcal{Q}_{\mathcal{O}}| < 2^{N_i+1}/10N_i^{1/\delta_1} \leq 2^{N_i+1-\log N_{\mathsf{long}}}$, this is possible. Then we embed x_{hard} into $\mathcal{O}$: we set

$$\mathcal{O}[\mathsf{pfx}_{x_{\mathsf{hard}}} \circ j] = (x_{\mathsf{hard}})_j, \forall j \in \{0,1\}^{\log N_{\mathsf{long}}}.$$

We also need to embed every truth table in $\mathcal{Q}_{\mathsf{useful}}$ into $\mathcal{O}$. For every such truth table $x \in \{0,1\}^{N_{\mathsf{short}}}$, we find a (distinct) prefix string pfx_x of length $N_i + 1 - \log N_{\mathsf{short}}$, and set

$$\mathcal{O}[\mathsf{pfx}_x \circ j] = x_j, \forall j \in \{0,1\}^{\log N_{\mathsf{short}}}.$$

Of course, we can use prefixes pfx_x only if (1) pfx_x is not the prefix of any string in $\mathcal{Q}_{\mathcal{O}}$, and (2) $\mathsf{pfx}_{x_{\mathsf{hard}}}$ is not a prefix of pfx_x. The number of available prefixes is at least

$$2^{N_i+1-\log N_{\mathsf{short}}} - |\mathcal{Q}_{\mathcal{O}}| - \frac{N_{\mathsf{long}}}{N_{\mathsf{short}}} \geq \Omega(2^{N_i}/N_{\mathsf{short}}),$$

which is greater than $|\mathcal{Q}_{\mathsf{useful}}| \leq 2^{N_i}/N_i^{\omega(1)}$. Therefore we can assign one prefix to each truth table $x \in \mathcal{Q}_{\mathsf{useful}}$.

Case II: M_i outputs 1 on x_{hard}. In this case, we want the circuit complexity of x_{hard} to be at least $N_i + 2$. By the choice of x_{hard}, it suffices to construct the length-$(N_i + 1)$ slice of $\mathcal{O}$ such that any single-$\mathcal{O}$-gate circuit of size $N_i + 1$ cannot compute x_{hard}.

Our oracle $\mathcal{O}$ will be consistent with $\mathcal{O}_{\mathsf{noise}}$, thus by Lemma 3.6, the circuit complexity of x_{hard} is indeed greater than $N_i + 1$. Again, for every truth table $x \in \mathcal{Q}_{\mathsf{useful}}$, we assign it a prefix pfx_x of length $N_i + 1 - \log N_{\mathsf{short}}$, and set

$$\mathcal{O}[\mathsf{pfx}_x \circ j] = x_j, \forall j \in \{0,1\}^{\log N_{\mathsf{short}}}.$$

By Lemma 3.7, there are $\Omega(2^{N_i}/N_{\mathsf{short}})$ good prefixes. Some of them are prefixes of strings in $\mathcal{Q}_{\mathcal{O}}$, therefore we cannot use these strings. There are still

$$\Omega(2^{N_i}/N_{\mathsf{short}}) - |\mathcal{Q}_{\mathcal{O}}| = \Omega(2^{N_i}/N_{\mathsf{short}}) \geq |\mathcal{Q}_{\mathsf{useful}}|$$

prefixes we can use, therefore we can also embed every truth table in $\mathcal{Q}_{\mathsf{useful}}$ into the oracle $\mathcal{O}$.

Finale. Let N_{i+1} be the length of the longest currently-fixed input in $\mathcal{O}$. We fix every unfixed input of length at most N_{i+1} in $\mathcal{O}$ arbitrarily, and conclude stage $(i + 1)$. We can see that M_i does not compute $\mathrm{MCSP}[2^{\delta_1 n}]$ on input length N_{long}.

3.3 $\mathrm{MCSP}[2^{\delta_1 n}]$ vs. $\mathrm{MCSP}[2^{\delta_2 n}]$, with Respect to Zero-Error Average-Case Complexity

Let $0 < \delta_1 < \delta_2 < 1$ be two constants. With respect to zero-error average-case complexity, it is easy to see that $\mathrm{MCSP}[2^{\delta_1 n}]$ is no harder than $\mathrm{MCSP}[2^{\delta_2 n}]$ [HS17]: Any algorithm that solves $\mathrm{MCSP}[2^{\delta_2 n}]$ on average automatically solves $\mathrm{MCSP}[2^{\delta_1 n}]$ on average. In this section, we construct a relativized world such that there is a linear-time zero-error average-case heuristic for $\mathrm{MCSP}[2^{\delta_1 n}]$, but any zero-error average-case heuristic for $\mathrm{MCSP}[2^{\delta_2 n}]$ requires $2^{N^{\delta_2}}/N^{\omega(1)}$ time. Actually, there will be no useful property against $\mathsf{SIZE}[2^{\delta_2 n}]$ that is constructive in $2^{N^{\delta_2}}/N^{\omega(1)}$ time (*regardless of density*)! Again, our result is nearly tight: the brute force algorithm can solve $\mathrm{MCSP}[2^{\delta_2 n}]$ in $2^{\tilde{O}(N^{\delta_2})}$ time in the worst case, in any relativized world.

Like Section 3.1, this world also contains two oracles: $\mathcal{O}$ and itrMCSP. However, the difference is that there is a (hidden) subset of strings $\mathsf{Corrupt}$ that plays a role in the definition of itrMCSP. This set is sparse: for every length N, there are at most $o(2^N)$ strings in $\mathsf{Corrupt} \cap \{0,1\}^N$. The precise definition of itrMCSP is as follows:

Definition 3.8. The oracle itrMCSP receives two inputs: a nonnegative integer k which is the "iteration number", and a truth table x. *If* $x \in \mathsf{Corrupt}$, *then* itrMCSP *outputs* 1. Otherwise, it outputs 1 if and only if there is a circuit C of size at most $\lfloor |x|^{\delta_1} \rfloor$ with $\mathcal{O}$ and itrMCSP gates whose truth table is x, such that on every invocation of itrMCSP on every possible input of C, the iteration number (i.e. the first input parameter) is at most $k - 1$. (In particular, if $k = 0$, then we do not allow oracle access to itrMCSP.)

We can see that itrMCSP is completely determined by $\mathcal{O}$ and $\mathsf{Corrupt}$. We can also construct a linear time algorithm that solves $\mathrm{MCSP}[2^{\delta_1 n}]$ on average: On input $x \in \{0,1\}^N$, if $\mathrm{itrMCSP}[2^N, x] = 1$, the algorithm outputs "easy"; otherwise it outputs "hard". First, the probability that the algorithm outputs "hard" on a uniformly random input is at least

$$1 - o(1) - \frac{|\mathsf{Corrupt} \cap \{0,1\}^N|}{2^N} \geq 1 - o(1).$$

Second, for any string x such that $\mathsf{CC}^{\mathcal{O},\mathrm{itrMCSP}}(x) \leq \lfloor |x|^{\delta_1} \rfloor$, by definition of itrMCSP, the algorithm will output "easy".

Now we need to construct $\mathcal{O}$ and $\mathsf{Corrupt}$ carefully to diagonalize against every candidate constructive property against $\mathsf{SIZE}[2^{\delta_2 n}]$. Let $\{M_i\}_{i\in\mathbb{N}}$ be an enumeration of Turing machines that run in $2^{N^{\delta_2}}/N^{\omega(1)}$ time. Again, our construction proceeds in stages. At the end of each stage (say stage i), there will be finitely many inputs to $\mathcal{O}$ that are fixed, and finitely many strings that are in $\mathsf{Corrupt}$. We define N_i to be the smallest integer such that both of the following hold:

- Every fixed input to $\mathcal{O}$ has length at most N_i.
- Let N_{long} be the largest power of 2 such that $\lfloor (N_{\mathsf{long}})^{\delta_1} \rfloor \leq N_i$, then every string in $\mathsf{Corrupt}$ has length at most N_{long}.

At the beginning of stage $i+1$, we fix every input to $\mathcal{O}$ with length at most N_i. Also, every string of length at most N_{long} that are not in $\mathsf{Corrupt}$ yet will never be in $\mathsf{Corrupt}$. By the same reasoning as Claim 3.3, every query $\mathrm{itrMCSP}[k,x]$ with $|x| \leq N_{\mathsf{long}}$ (i.e. $\lfloor |x|^{\delta_1} \rfloor \leq N_i$) are fixed.

Now we describe stage $i+1$, in which we diagonalize against M_i. Let N_{short} be the smallest power of 2 such that $\lfloor (N_{\mathsf{short}})^{\delta_2} \rfloor \geq N_i + 1$. For every input $x \in \{0,1\}^{N_{\mathsf{short}}}$, we run M_i on input x.

- Whenever M_i asks a query $\mathcal{O}[x]$, if $\mathcal{O}[x]$ is already fixed then we return $\mathcal{O}[x]$; otherwise we return and fix $\mathcal{O}[x]$ to be 0.
- Whenever M_i asks a query $\mathrm{itrMCSP}[k,x]$, if $|x| \leq N_{\mathsf{long}}$, then we return the already-fixed $\mathrm{itrMCSP}[k,x]$; otherwise we return 1.

During these $2^{N_{\mathsf{short}}}$ invocations, M_i asked at most $2^{N_{\mathsf{short}}+(N_{\mathsf{short}})^{\delta_2}}$ queries (k,x) to the itrMCSP oracle. For every queried truth table x, if $|x| > N_{\mathsf{long}}$, then we add x into $\mathsf{Corrupt}$. It is easy to see that $\mathsf{Corrupt}$ is still a sparse set. Actually, since $\delta_2 > \delta_1$, for every input length $\ell > N_{\mathsf{long}}$,

$$\frac{|\mathsf{Corrupt} \cap \{0,1\}^\ell|}{2^\ell} \leq \frac{2^{N_{\mathsf{short}}+(N_{\mathsf{short}})^{\delta_2}}}{2^{N_{\mathsf{long}}}} = \frac{2^{O\left(N_i^{1/\delta_2}\right)}}{2^{\Omega\left(N_i^{1/\delta_1}\right)}} = o(1).$$

(Note that we have not added any string of length ℓ to $\mathsf{Corrupt}$ in previous stages; every such string is added in the current stage.)

Suppose M_i outputs "easy" on every input. Then we immediately finish stage $i+1$. We can see that M_i cannot be a constructive property against $\mathsf{SIZE}[2^{\delta_2 n}]$, since it outputs "easy" on every input of length N_{short}.

Now suppose M_i outputs "hard" on some input $x \in \{0,1\}^{N_{\mathsf{short}}}$. We run M_i on x, and let $\mathcal{Q}_{\mathcal{O}}$ be the set of all queries made to $\mathcal{O}$ with length exactly N_i+1, during the execution of $M_i(x)$. We will "embed" x into the length-(N_i+1) slice of $\mathcal{O}$, such that $\mathsf{CC}^{\mathcal{O}}(x) \leq N_i+1 \leq \lfloor (N_{\mathsf{short}})^{\delta_2} \rfloor$. Note that this would affect the execution of M_i on other inputs than x, but as M_i already outputs "hard" on the easy input x, it is not a constructive property against $\mathsf{SIZE}[2^{\delta_2 n}]$ anyway. Since M_i runs in $2^{(N_{\mathsf{short}})^{\delta_2}}/(N_{\mathsf{short}})^{\omega(1)} < 2^{N_i - \log|N_{\mathsf{short}}|}$ time, we have $|\mathcal{Q}_{\mathcal{O}}| < 2^{N_i+1-\log|N_{\mathsf{short}}|}$. Fix an arbitrary prefix pfx of length $N_i + 1 - \log|N_{\mathsf{short}}|$ which is not a prefix of any string in $\mathcal{Q}_{\mathcal{O}}$, and embed x as follows:

$$\mathcal{O}[\mathsf{pfx} \circ j] = x_j, \forall j \in \{0,1\}^{\log N_{\mathsf{short}}}.$$

We can finish stage $i+1$ now. As M_i outputs "hard" on the easy input x, it is not a constructive property.

We have the following corollary that in some relativized worlds, the worst-case complexity and average-case complexity of $\mathrm{MCSP}[2^{\epsilon n}]$ can be very different:

Corollary 3.9. *Let $0 < \epsilon < 1/2$ be a constant. Then there is an oracle relative to which* $\mathrm{MCSP}[2^{\epsilon n}]$ *has a linear time zero-error heuristic, but the same problem* $\mathrm{MCSP}[2^{\epsilon n}]$ *cannot be solved deterministically in* $2^{N^\epsilon}/N^{\omega(1)}$ *time in the worst case.*

Proof. Consider the oracle in which $\mathrm{MCSP}[2^{\epsilon n}]$ has a linear time zero-error heuristic, but there is no useful property against $\mathsf{SIZE}[2^{2\epsilon n}]$ that is constructive in $2^{N^{2\epsilon}}/N^{\omega(1)}$ time. Suppose, for the sake of contradiction, that $\mathrm{MCSP}[2^{\epsilon n}]$ can be solved in $2^{N^\epsilon}/N^{\omega(1)}$ time in the worst case.

Let x be a length-N input to $\mathrm{MCSP}[2^{2\epsilon n}]$, and y be the concatenation of N copies of x. Since the truth table y only depends on its last $\log N$ bits, we have $\mathsf{CC}(y) = \mathsf{CC}(x)$. Therefore

$$y \in \mathrm{MCSP}[2^{\epsilon n}] \iff \mathsf{CC}(y) \le (N^2)^\epsilon \iff \mathsf{CC}(x) \le N^{2\epsilon} \iff x \in \mathrm{MCSP}[2^{2\epsilon n}].$$

That is, using the algorithm for $\mathrm{MCSP}[2^{\epsilon n}]$, we can solve $\mathrm{MCSP}[2^{2\epsilon n}]$ in the worst case in $2^{N^{2\epsilon}}/N^{\omega(1)}$ time, which implies a (pretty dense) natural property against $\mathsf{SIZE}[2^{2\epsilon n}]$. □

4 A World Where GapMKtP Is Easy

In this section, we construct a relativized world where Levin's Kt complexity can be approximated within a multiplicative factor of $2 + o(1)$ in linear time. Actually, we will introduce a non-standard variant of Kt complexity, which we call $\widetilde{\mathrm{Kt}}$, such that $\widetilde{\mathrm{Kt}}$ is computable *exactly* in linear time, and for every string x, $\widetilde{\mathrm{Kt}}(x) \le \mathrm{Kt}(x) \le (2 + o(1))\widetilde{\mathrm{Kt}}(x)$. By the properties of $\widetilde{\mathrm{Kt}}$, it follows directly that the Kt complexity is easy to $(2 + o(1))$-approximate in this relativized world.

The computational model we consider is Turing machines with (say) 3 working tapes. In oracle worlds, one of these tapes is a query tape. For every oracle $\mathcal{O}$ in the oracle world, the Turing machine has two special states: a query state $s^{\mathcal{O}}_{\mathsf{query}}$ and a state $s^{\mathcal{O}}_{\mathsf{afterquery}}$ indicating a query has just finished. Upon entering $s^{\mathcal{O}}_{\mathsf{query}}$, we will treat the content of the query tape as the query to $\mathcal{O}$. After one step, the content of the query tape becomes the answer to this query, and we enter the state $s^{\mathcal{O}}_{\mathsf{afterquery}}$.

For a string x, we define $\widetilde{\mathrm{Kt}}(x)$ to be the minimum of $|M| + \lfloor \log t \rfloor$ over all Turing machine M and time bound t, where after running M for t steps, the content of some tape is exactly x. Note that we do *not* require M to terminate after t steps, which is the difference between our definition of $\widetilde{\mathrm{Kt}}$ and the standard definition of Kt.

We first show that $\widetilde{\mathrm{Kt}}$ approximates Kt well.

Lemma 4.1. *For every string x, we have* $\widetilde{\mathrm{Kt}}(x) \le \mathrm{Kt}(x) \le (2 + o(1))\widetilde{\mathrm{Kt}}(x)$.

Proof. It is easy to see that $\widetilde{\mathrm{Kt}}(x) \le \mathrm{Kt}(x)$, so we proceed to show that $\mathrm{Kt}(x) \le (2+o(1))\widetilde{\mathrm{Kt}}(x)$. Suppose $\widetilde{\mathrm{Kt}}(x) = |M| + \lfloor \log t \rfloor$, where M is a Turing machine such that after running M for t steps, the content of some tape is exactly x. We hardcode t into the machine M, and force M to terminate in t steps (which costs $\log t$ description length). We also hardcode the index of the tape of M that contains x (which only costs $O(1)$ length), so upon termination M outputs the contents on this tape. Thus we obtain a machine with description length $|M| + \log t + O(1)$ that terminates in $O(t)$ steps, and has x as the output. It follows that

$$\begin{aligned}\mathrm{Kt}(x) &\le |M| + \log t + O(1) + \log O(t)\\ &\le |M| + 2\log t + O(1)\\ &\le (2 + o(1))\widetilde{\mathrm{Kt}}(x).\end{aligned}$$

□

It is easy to see that the above proof relativizes, i.e. for every oracle $\mathcal{O}$, $\widetilde{\mathrm{Kt}}^{\mathcal{O}}(x) \le \mathrm{Kt}^{\mathcal{O}}(x) \le (2 + o(1))\widetilde{\mathrm{Kt}}^{\mathcal{O}}(x)$.

Now we prove our main theorem.

Theorem 4.2. *There is an oracle $\mathcal{O}$ relative to which $\widetilde{\mathrm{Kt}}$ is computable in linear time.*

Proof. We will construct a "fixed-point" oracle $\mathcal{O}$ such that for every string x, $\mathcal{O}[x] = \widetilde{\mathrm{Kt}}^{\mathcal{O}}(x)$. (That is, $\mathcal{O}[x]$ is exactly the $\widetilde{\mathrm{Kt}}$ complexity of x in a world with $\mathcal{O}$ itself as an oracle.)

The construction is divided into stages, and in stage n, we will fix every input on which $\mathcal{O}$ outputs n. In this stage, we enumerate every pair (M, t) such that $|M| + \lfloor \log t \rfloor = n$. Then for each such (M, t), we run M for t steps. Whenever it makes a query x to $\mathcal{O}$, if we have already fixed $\mathcal{O}[x]$, then we simply return $\mathcal{O}[x]$; otherwise we fix $\mathcal{O}[x] = n$ and return n. Finally, for each tape of M with content y, if $\mathcal{O}[y]$ is not fixed, then we fix $\mathcal{O}[y] = n$.

First we show that for every string x, $\widetilde{\mathrm{Kt}}^{\mathcal{O}}(x) \leq \mathcal{O}[x]$. Suppose $\mathcal{O}[x] = n$ is fixed when we run the machine M with time bound t, then $|M| + \lfloor \log t \rfloor = n$. If x is an oracle query asked by M at time t', then $\widetilde{\mathrm{Kt}}^{\mathcal{O}}(x) \leq |M| + \lfloor \log t' \rfloor \leq |M| + \lfloor \log t \rfloor = n$. On the other hand, if $\mathcal{O}[x]$ is fixed when we finish running M, then we also have $\widetilde{\mathrm{Kt}}^{\mathcal{O}}(x) \leq n$.

Then we show that for every string x, $\mathcal{O}[x] \leq \widetilde{\mathrm{Kt}}^{\mathcal{O}}(x)$. Suppose that $\widetilde{\mathrm{Kt}}^{\mathcal{O}}(x) = n$, as witnessed by a Turing machine M and a time bound t where $|M| + \lfloor \log t \rfloor = n$. Moreover, M is a Turing machine such that after running M for t steps, the content of some tape is exactly x. At stage n, after running M for t steps, if $\mathcal{O}[x]$ is not already fixed (to a potentially smaller value), we will always set $\mathcal{O}[x]$ to be equal to n. Thus $\mathcal{O}[x] \leq \widetilde{\mathrm{Kt}}^{\mathcal{O}}(x)$. □

4.1 Extension: A World Where $\widetilde{\mathrm{Kt}}$ Is Easy and EXP = ZPP

The proof that MKtP is EXP-complete under $\mathsf{P}_{/\mathrm{poly}}$-truth-table reductions and NP-Turing reductions [ABK+06] already uses *non-relativizing* techniques (namely, an EXP-complete problem with instance checkers [BFL91, BFNW93]). Therefore, the current *relativization* barrier seems unsatisfactory. A natural direction would be to present an *algebrization* barrier [AW09], but currently we are unable to do so. Nevertheless, we show that there is a relativized world where $\widetilde{\mathrm{Kt}}$ is computable in polynomial time, and EXP = ZPP.[6]

Our world has two oracles: $\mathcal{O}$ and Cheat. The oracle $\mathcal{O}$ will compute $\widetilde{\mathrm{Kt}}$ exactly, i.e. for every string x, $\mathcal{O}[x] = \widetilde{\mathrm{Kt}}^{\mathcal{O},\mathsf{Cheat}}(x)$. The oracle Cheat will help a ZPP machine (trying to solve an EXP problem) to "cheat", by storing the truth table of some EXP-complete problem.

We proceed in stages. Let n be a positive integer. In the n-th stage, we assume that every input to Cheat with length at most $3(n-1)$ are fixed. At the end of this stage, we will fix the inputs to Cheat with length at most $3n$, and potentially a few longer inputs. In this stage, we examine every pair (M, t), where M is a Turing machine, t is a time bound, and $|M| + \lfloor \log t \rfloor = n$. We run M for t steps.

- Each time M makes a query $\mathcal{O}[x]$, if $\mathcal{O}[x]$ is already fixed, then we return $\mathcal{O}[x]$; otherwise, we fix and return $\mathcal{O}[x] = n$.
- Each time M makes a query $\mathsf{Cheat}[x]$, if $\mathsf{Cheat}[x]$ is already fixed, then we return $\mathsf{Cheat}[x]$; otherwise we fix and return $\mathsf{Cheat}[x] = 0$.
- Finally, for each tape of M at the t-th step, suppose its content is x, then we fix $\mathcal{O}[x] = n$.

Now, the construction of oracle $\mathcal{O}$ in stage n is done. However, we still need to construct the Cheat oracle, so that EXP = ZPP in our relativized world. This is done by filling the Cheat oracle with a string tt^n that captures the truth table of some E-complete problem.

The precise definition of tt^n is as follows. Let $\langle \cdot, \cdot \rangle$ be a bijection from the set of (M, t)'s such that $|M| + \lfloor \log t \rfloor = n$ to the set $[n \cdot 2^n]$. For example, let M be a length-ℓ Turing machine, and

[6] We also point out that in the unrelativized world, EXP = ZPP if and only if $R_{\mathrm{Kt}} = \{x : \mathrm{Kt}(x) \geq |x|/3\}$ is in ZPP (see [ABK+06]). The proof uses IP = PSPACE and thus does not relativize.

t be an integer in $[2^{n-\ell}, 2^{n-\ell+1})$, then $\langle M, t\rangle = (\ell-1)\cdot 2^n + M \cdot 2^{n-\ell} + (t - 2^{n-\ell})$, where M is naturally interpreted as an integer in $[0, 2^\ell)$. For every (M, t) such that $|M| + \lfloor \log t \rfloor = n$, if M outputs exactly one bit b after t steps, then the $\langle M, t\rangle$-th bit of tt^n is b; otherwise the $\langle M, t\rangle$-th bit of tt^n is (say) 0. Finally, we artificially define the $(n \cdot 2^n)$-th bit of tt^n as 1 (i.e. append a bit 1 to the end of tt^n). By this definition, tt^n is a string of length $n \cdot 2^n + 1$, and given access to $tt^{O(n)}$, we can solve an E-complete problem on input length n easily.

Now we put tt^n into the length-$3n$ slice of the Cheat oracle. Before we do that, we argue that we have enough space to put tt^n (in fact, a lot of copies of tt^n) into Cheat. Actually, in the first n stages, we enumerate every (M, t) where M is a Turing machine, and $|M| + \lfloor \log t \rfloor \le n$. These Turing machines make queries to Cheat; let $\mathcal{Q}_{3n}$ be the set of all queries these machines make to the $3n$-th slice of Cheat. Then

$$|\mathcal{Q}_{3n}| \le \sum_{M:|M|\le n} 2^{n-|M|} = \sum_{\ell=1}^{n} 2^\ell \cdot 2^{n-\ell} = n \cdot 2^n.$$

Let pfx be any string ("prefix") of length $3n - \lceil \log(n \cdot 2^n + 1)\rceil$. We say pfx is *clear* if there is no string $q \in \mathcal{Q}_{3n}$ such that pfx is a prefix of q. The number of clear prefixes is at least $2^{|\mathsf{pfx}|} - n \cdot 2^n$, since each string in $\mathcal{Q}_{3n}$ only "kills" at most one prefix. For every clear prefix pfx, and every index $0 \le j \le n \cdot 2^n$, we fix

$$\mathsf{Cheat}[\mathsf{pfx} \circ j] = (tt^n)_j, \tag{3}$$

where $\mathsf{pfx} \circ j$ is the concatenation of pfx and j, and j is interpreted as a string of length $\lceil \log(n2^n + 1)\rceil$.

Finally, for every input to Cheat that is currently unfixed and has length at most $3n$, we fix it to 0. Note that there is an easy way to distinguish whether a prefix pfx is clear or not: let $j = n \cdot 2^n + 1$, then pfx is clear if and only if $\mathsf{Cheat}[\mathsf{pfx} \circ j] = 1$. This is because 1) we artificially appended a bit 1 to the end of tt^n, and 2) every query in $\mathcal{Q}_{3n}$ and every query not fixed by Eq. (3) are fixed to 0. This concludes the description of stage n.

Now we prove the desired properties of this oracle world.

Lemma 4.3. *For every string x, $\widetilde{\mathrm{Kt}}^{\mathcal{O},\mathsf{Cheat}}(x) = \mathcal{O}[x]$.*

Proof Sketch. We use the same proof as in Theorem 4.2. In particular, every time we fix a query $\mathcal{O}[x] = n$, we already have a "witness" that $\widetilde{\mathrm{Kt}}^{\mathcal{O},\mathsf{Cheat}}(x) \le n$. On the other hand, every $\mathcal{O}[x]$ is fixed when we run the optimal machine for x. □

Lemma 4.4. $\mathsf{EXP}^{\mathcal{O},\mathsf{Cheat}} \subseteq \mathsf{ZPP}^{\mathsf{Cheat}}$.

Proof. Let M be a Turing machine of length ℓ that outputs one bit in $t = 2^\ell$ steps. We want to compute the output of M in $\mathsf{ZPP}^{\mathsf{Cheat}}$.

Let $n = 2\ell$. First, we sample a clear prefix pfx. Over all strings pfx of length $3n - \lceil \log(n \cdot 2^n + 1)\rceil$, there are only $n \cdot 2^n$ prefixes that are not clear, so a vast majority of prefixes are clear. To check whether pfx is a clear prefix, we let $j = n \cdot 2^n + 1$, and check whether $\mathsf{Cheat}[\mathsf{pfx} \circ j] = 1$. Therefore we can sample a clear prefix pfx in ZPP.

Then, by Eq. (3), the output of M is exactly $\mathsf{Cheat}[\mathsf{pfx} \circ \langle M, t\rangle]$. Therefore, we can simulate any $\mathsf{EXP}^{\mathcal{O},\mathsf{Cheat}}$ machine in $\mathsf{ZPP}^{\mathsf{Cheat}}$. □

4.2 More on $\widetilde{\mathrm{Kt}}$

We think the notion of $\widetilde{\mathrm{Kt}}$ complexity is of independent interest. In this section, we prove a preliminary result and pose some open questions about $\widetilde{\mathrm{Kt}}$.

First, a random string has high $\widetilde{\mathrm{Kt}}$ complexity.

Fact 4.5. *For every integer n, w.p. at least $1/2$, a random string x of length n has $\widetilde{\mathrm{Kt}}$ complexity at least $n - \log n - 1$.*

Proof. Let $k = n - \log n - 2$, we use a counting argument to show that there are at most $2^n/2$ strings with $\widetilde{\mathrm{Kt}}$ complexity at most k. For every integer $\ell \in [1, k]$, there are 2^ℓ Turing machines of description length ℓ. Each of these Turing machines, when given $2^{k-\ell+1}$ time, can generate at most $2^{k-\ell+1}$ strings. (Note that in each step, the content of only one tape may change.) Thus, the number of strings with $\widetilde{\mathrm{Kt}}$ complexity at most k that are contributed by some length-ℓ Turing machine is at most

$$2^\ell \cdot 2^{k-\ell+1} \leq 2^{k+1}.$$

To upper bound the number of strings with $\widetilde{\mathrm{Kt}}$ complexity at most k, we sum over every possible $\ell \in [1, k]$. Therefore, there are at most

$$2^{k+1} \cdot k \leq 2^{n-\log n-1} \cdot n \leq 2^n/2$$

strings with $\widetilde{\mathrm{Kt}}$ complexity at most k. □

However, we do not know whether Fact 4.5 is tight.

Open Problem 4.6. *Is Fact 4.5 tight? For large enough n, are there length-n strings with $\widetilde{\mathrm{Kt}}$ complexity at least $n - 100$?*

It seems that Kt and $\widetilde{\mathrm{Kt}}$ should be two completely different complexity measures. However, we do not know whether they are provably different (in the unrelativized world).

Open Problem 4.7. *Are there infinitely many strings x such that $\mathrm{Kt}(x) \neq \widetilde{\mathrm{Kt}}(x)$? We conjecture the answer is yes, but we do not have a proof yet. A more difficult question would be: Is there a long-enough string x such that $\mathrm{Kt}(x) \geq 1.9\widetilde{\mathrm{Kt}}(x)$?*

Finally, consider the oracle $\mathcal{O}$ constructed in the proof of Theorem 4.2. Is this oracle $\mathcal{O}$ the same oracle as the unrelativized $\widetilde{\mathrm{Kt}}$, or are they different functions? We conjecture they are different, but be careful: *this conjecture needs nonrelativizing techniques to prove*!

Open Problem 4.8. *Is the oracle $\mathcal{O}$ constructed in Theorem 4.2 different from the unrelativized $\widetilde{\mathrm{Kt}}$ function?*

5 Finding More Pessilands

The word *Pessiland*, coined by Impagliazzo [Imp95], refers to a world where NP contains a language that is hard on average, yet no one-way functions exist. In this section, we present Pessilands where not only one-way functions, but also auxiliary-input one-way functions (AIOWF; see Definition 2.9) do not exist. Moreover, meta-complexity problems are hard on average in our Pessilands.

Theorem 5.1. *For each of the following items, there is a relativized world where the item is true, yet auxiliary-input one-way functions do not exist.*

- *Let $\delta \in (0, 1)$ be a constant. There is no $\mathsf{SIZE}[2^{o(N^\delta)}]$-natural property that is useful against $\mathsf{SIZE}[2^{\delta n}]$ and has density $2^{-o(N^\delta)}$.*
- *There is a polynomial-time samplable distribution under which no circuit of $2^{o(N)}$ size solves* MCSP *with error probability < 0.01.*
- *There is a polynomial-time samplable distribution under which no circuit of $2^{o(N)}$ size solves* MINKT *with error probability < 0.01.*

Our techniques build on the paper titled "Finding Pessilands" by [Wee06], hence we name this section as "Finding More Pessilands".

5.1 The Construction

Our Pessilands follow the same template. We will have a "base oracle" Π, and an oracle f_{PSPACE} which is a PSPACE^{Π}-complete oracle. For example, f_{PSPACE} may be the following oracle:

- Given a Turing machine M that has access to Π oracles, and an integer s represented in unary, f_{PSPACE} outputs 1 if M accepts the empty input within space complexity s, and outputs 0 otherwise.

However, f_{PSPACE} does not need to be exactly the problem above. For example, in Section 5.3, we will add some padding to the f_{PSPACE} oracle. What is important is that f_{PSPACE} should be PSPACE^{Π}-complete.

For every input length n, we also have a function $f_n : \{0,1\}^n \to \{0,1\}^n$. For intuition, we can think of the functions $\{f_n\}_{n\in\mathbb{N}}$ as chosen uniformly at random. More precisely, we will choose f_n to satisfy the following constraint:

Constraint 5.2. For every integer s, conditioned on the oracles Π and f_{PSPACE} on input lengths at most s, the functions $\{f_n\}_{n=1}^{s}$ cannot be described in less than $N - 5\log N$ bits, where $N = \sum_{n=1}^{s} n2^n$.

We will also have an oracle $\mathsf{Trapdoor}$, which receives three inputs x, y, β, where $|x| = |y| = n$ and $|\beta| = \lceil c \log n \rceil$.[7] (Here, c is a constant to be fixed.) The definition of $\mathsf{Trapdoor}$ is as follows:

- Given inputs x, y, β, if $y \neq f_{|x|}(x)$, we define $\mathsf{Trapdoor}[x, y, \beta] = \perp$; otherwise we define $\mathsf{Trapdoor}[x, y, \beta]$ in some way, possibly depending on Π.

The above illustrates the basic framework of our Pessiland constructions. In this framework, we have *not* specified the following items, and different specifications of these items lead to different relativized worlds (many of which are Pessilands):

- The oracle Π.
- The exact definition of f_{PSPACE} (we only need it to be PSPACE^{Π}-complete).
- The exact definition of $\mathsf{Trapdoor}[x, y, \beta]$ in case that $y = f_{|x|}(x)$. (Again, we stress that this definition may depend on Π.)

Note that we do not take the functions $\{f_n\}_{n\in\mathbb{N}}$ into the "degree of freedom": any choice of $\{f_n\}$ satisfying Constraint 5.2 will work.

Even though the above items are not defined yet, we can already prove that there are no auxiliary-input one-way functions in this oracle world, unless $\{f_n\}$ are not (Kolmogorov-)random.

The proof consists of two steps. In the first step, based on the (Kolmogorov-)randomness of $\{f_n\}_{n\in\mathbb{N}}$, we show that every circuit can be approximated by a slightly larger circuit without $\mathsf{Trapdoor}$ gates. More precisely:

Lemma 5.3. *Let s be an integer. For every circuit $C : \{0,1\}^n \to \{0,1\}^m$ of size s and every distribution $\mathcal{D}$ over $\{0,1\}^n$ that can be described in $O(\log n)$ bits, there is a circuit C' of $\mathrm{poly}(s)$ size without $\mathsf{Trapdoor}$ gates, such that*

$$\Pr_{\mathbf{z}\leftarrow\mathcal{D}}[C(\mathbf{z}) \neq C'(\mathbf{z})] \leq 1/s.$$

In the second step, we use the f_{PSPACE} oracle and Lemma 5.3 to invert every AIOWF. Roughly speaking, given a circuit C that implements a candidate OWF, we use Lemma 5.3 to find a circuit C' without $\mathsf{Trapdoor}$ gates that is close to C. Inverting C' can be done in polynomial space with a Π oracle, therefore it can also be done in polynomial time with a PSPACE^{Π}-complete oracle.

Lemma 5.4. *There are no auxiliary-input one-way functions in our oracle worlds.*

As the proofs are long and technical, we leave them in Section 5.5 and 5.6 respectively.

[7] The name "$\mathsf{Trapdoor}$" will be justified in Section 5.3.

5.2 Pessiland I: Ruling Out Natural Proofs

With the above framework, we can construct a relativized world where both AIOWFs and natural proofs do not exist. We already ruled out AIOWFs in Lemma 5.4, and here we prove that our world does not have $\mathsf{SIZE}[2^{o(N^\delta)}]$-constructive natural proofs that is useful against $\mathsf{SIZE}[O(2^{\delta n})]$, and has density $2^{-o(N^\delta)}$. Here we set $c = 1/\delta$. (That is, for a valid input (x, y, β) to the $\mathsf{Trapdoor}$ gate, we have $|\beta| = \lceil \log |x|/\delta \rceil$.)

Let $\Pi = \varnothing$ be a trivial oracle, and f_{PSPACE} be the canonical PSPACE-complete oracle. We will use a pseudorandom distribution to define $\mathsf{Trapdoor}[x, y, \beta]$ (for every $y = f_{|x|}(x)$), as follows.

Consider a circuit $C : \{0,1\}^n \to \{0,1\}$ and a parameter $\epsilon > 0$, we say a multiset $\mathcal{S} \subseteq \{0,1\}^n$ *ϵ-fools* C if

$$\left| \Pr_{\mathbf{x} \leftarrow \mathcal{S}}[C(\mathbf{x}) = 1] - \Pr_{\mathbf{x} \leftarrow \mathcal{U}_n}[C(\mathbf{x}) = 1] \right| \leq \epsilon.$$

Here, $\mathbf{x} \leftarrow \mathcal{S}$ means that $\mathbf{x}$ is drawn from the uniform distribution over (the multiset) $\mathcal{S}$.

Now let N be a power of 2. A standard probabilistic argument (see, e.g. [Vad12, Proposition 7.8]) shows that there is a multiset $\mathcal{S} \subseteq \{0,1\}^N$ of size 2^{4N^δ} that $(1/2^{N^\delta})$-fools every circuit of size 2^{N^δ} with f_{PSPACE} oracle gates. We define $\mathcal{S}_N$ as the lexicographically smallest such multiset (w.r.t. some encoding) of size exactly 2^{4N^δ}.

Now we can define $\mathsf{Trapdoor}[x, y, \beta]$. For every $N = 2^n$, integer $x \in [0, 2^{4N^\delta})$, and index $\beta \in \{0,1\}^n$, we define $\mathsf{Trapdoor}[x, f_{|x|}(x), \beta]$ as the β-th bit of the x-th string in $\mathcal{S}_N$. For all other cases, we define $\mathsf{Trapdoor}[x, y, \beta] = \bot$. We can see that every string in $\mathcal{S}_N$ has circuit complexity at most $O(N^\delta)$ in this world.

Theorem 5.5. *In our Pessiland I, there are no $\mathsf{SIZE}[2^{o(N^\delta)}]$-natural properties that is useful against $\mathsf{SIZE}[O(2^{\delta n})]$ and has density $2^{-o(N^\delta)}$.*

Proof. Let $N = 2^n$ and $s = 2^{o(N^\delta)}$. Consider the following "hybrid" distribution $\mathcal{D}_{\mathsf{hyb}}$ over $\{0,1\}^N$: To sample from $\mathcal{D}_{\mathsf{hyb}}$, with probability $1/2$ we sample a uniformly random string from $\mathcal{U}_N$, and with probability $1/2$ we sample a uniformly random string from $\mathcal{S}_N$. Let C be a circuit of size s with f_{PSPACE} and $\mathsf{Trapdoor}$ gates. Suppose C is a natural proof useful against $\mathsf{SIZE}[2^{\delta n}]$. Note that $\mathcal{S}_N$, thus $\mathcal{D}_{\mathsf{hyb}}$, can be described by $O(\log N)$ bits. By Lemma 5.3, there is a circuit C' of size $\mathrm{poly}(s)$, that does not use $\mathsf{Trapdoor}$ gates, and satisfies

$$\Pr_{\mathbf{z} \leftarrow \mathcal{D}_{\mathsf{hyb}}}[C(\mathbf{z}) \neq C'(\mathbf{z})] \leq 1/s.$$

By definition of $\mathcal{D}_{\mathsf{hyb}}$, we have

$$\Pr_{\mathbf{z} \leftarrow \mathcal{U}_N}[C(\mathbf{z}) \neq C'(\mathbf{z})] \leq 2/s, \text{ and } \Pr_{\mathbf{z} \leftarrow \mathcal{S}_N}[C(\mathbf{z}) \neq C'(\mathbf{z})] \leq 2/s.$$

Now, for a distribution $\mathcal{D}$ and a circuit $C^\star$, let $C^\star(\mathcal{D})$ denote the probability that $C^\star(\mathbf{z}) = 1$ when $\mathbf{z}$ is sampled from $\mathcal{D}$. Then we have

$$|C(\mathcal{U}_N) - C'(\mathcal{U}_N)| \leq 2/s, \text{ and } |C(\mathcal{S}_N) - C'(\mathcal{S}_N)| \leq 2/s.$$

Since C' is a circuit of size $2^{o(N^\delta)}$ with only f_{PSPACE} oracle gates, we have $\mathcal{S}_N$ $(1/s)$-fools C'. This means

$$|C'(\mathcal{U}_N) - C'(\mathcal{S}_N)| \leq 1/s.$$

It follows that

$$|C(\mathcal{U}_N) - C(\mathcal{S}_N)| \leq 5/s.$$

However, since C is a natural proof useful against $\mathsf{SIZE}[O(2^{\delta n})]$, and every string in $\mathcal{S}_N$ has circuit complexity at most $O(2^{\delta n})$, we have that $C(\mathcal{S}_N) = 0$. It follows that $C(\mathcal{U}_N) \leq 5/s$, contradicting the density of C. □

Finally, we use an incompressibility argument to instantiate our Pessiland I. Let ω be an infinite binary string as in Theorem 2.13, such that for every large enough integer n, $\mathrm{C}(\omega_{1\sim n} \mid n) \geq n - 2\log n$. Let $\ell_n = n2^n$ which is the number of bits needed to encode the function f_n. Let $\ell_{\leq n} = \sum_{i=1}^{n} \ell_i$, and we define f_n as the function decoded in some canonical way from

$$\omega_{(\ell_{\leq n-1}+1)\sim \ell_{\leq n}}.$$

By definition of ω, Constraint 5.2 is satisfied. We have completed our description of Pessiland I by specifying $\{f_n\}_{n\in\mathbb{N}}$, Π (an empty oracle), f_{PSPACE} (a canonical PSPACE-complete oracle), and $\mathsf{Trapdoor}[x, f_{|x|}(x), \beta]$ (as above). By Lemma 5.4 and Theorem 5.5, there are neither AIOWFs nor natural proofs in our Pessiland I.

5.3 Pessiland II: MCSP Is Hard On Average

Our Pessiland II is a combination of the two Pessilands in [Wee06]. (Section 5.1 is adapted from the second Pessiland of [Wee06], and we add some elements in the first Pessiland of [Wee06] in this section.) We are able to show that MCSP is 0.01-hard under some samplable distribution. (Note that in Pessiland I, MCSP is hard under the distribution $\mathcal{D}_{\mathsf{hyb}}$, but $\mathcal{D}_{\mathsf{hyb}}$ is not necessarily efficiently samplable.)

Let c be a very large constant (in the definition of $\mathsf{Trapdoor}$). Besides the functions $\{f_n\}_{n\in\mathbb{N}}$, for every string $y \in \{0,1\}^n$, we will also have a permutation $\pi_y : \{0,1\}^{\lceil c\log n\rceil} \to \{0,1\}^{\lceil c\log n\rceil}$. We can also think of each π_y as independent and uniformly random variables; below is a rigorous definition from Kolmogorov-randomness.

Definition 5.6. Again, let ω be an infinite binary string satisfying the promise of Theorem 2.13. That is, for almost every integer n, $\mathrm{C}(\omega_{1\sim n} \mid n) \geq n - 2\log n$. Let

$$\ell_n = n2^n + \lfloor 2^n \log((2^{\lceil c\log n\rceil})!)\rfloor,$$

then we can encode the function f_n and every permutation π_y where $y \in \{0,1\}^n$ in $\ell_n + 1$ bits. We will take some ℓ_n bits from ω and decode from it the pair $\big(f_n, \{\pi_y\}_{y\in\{0,1\}^n}\big)$. (We cannot take $\ell_n + 1$ bits from ω since some string of length $\ell_n + 1$ may not correspond to any such pairs; but taking ℓ_n bits is okay.) Fix an injection Decode that maps a string $\{0,1\}^{\ell_n}$ to a pair $\big(f_n, \{\pi_y\}_{y\in\{0,1\}^n}\big)$, such that both Decode and its inverse Decode^{-1} are efficiently computable. Let $\ell_{\leq n} = \sum_{i=1}^{n} \ell_i$, then $\ell_{\leq n} \leq 3^n$ for large enough n. We define

$$\big(f_n, \{\pi_y\}_{y\in\{0,1\}^n}\big) = \mathsf{Decode}\Big(\omega_{(\ell_{\leq n-1}+1)\sim \ell_{\leq n}}\Big).$$

That is, we decode the function f_n and permutations $\{\pi_y\}_{y\in\{0,1\}^n}$ from the corresponding segment of ω.

Oracles. Given $\{f_n\}$ and $\{\pi_y\}$, we instantiate Pessiland II as follows:

- On inputs y and α, $\Pi[y, \alpha]$ returns $\pi_y(\alpha)$.
- On inputs $x \in \{0,1\}^n, y \in \{0,1\}^n$, and $\beta \in \{0,1\}^{\lceil c\log n\rceil}$, if $f_n(x) = y$, then $\mathsf{Trapdoor}[x, y, \beta] = \pi_y^{-1}(\beta)$; otherwise $\mathsf{Trapdoor}[x, y, \beta] = \bot$.
- We add suitable padding to the oracle f_{PSPACE}, so it does not help too much for inverting permutations π_y. More precisely, this oracle receives inputs $M, 1^s$ and $1^{(|M|+s)^c}$, where M is a Turing machine with oracle access to Π (but not $\mathsf{Trapdoor}$), and s is a space bound. It returns 1 if M accepts the empty input within space complexity s, and returns 0 otherwise. (We emphasize that every query of M to Π also has length at most s.)

(We call $\mathsf{Trapdoor}$ a "trapdoor" oracle because given a string y, if we know a "key" x such that $f_{|x|}(x) = y$, then the "trapdoor" oracle allows us to invert π_y faster than brute force. Our hardness result for MCSP will exploit this fact.)

We can see that Constraint 5.2 is satisfied. Indeed, suppose that conditioned on the oracles Π and f_{PSPACE} on input lengths at most s, we can describe $\{f_n\}_{n=1}^s$ in less than $N' - 5\log N'$ bits, where $N' = \sum_{n=1}^s n2^n$. Now let $N = \ell_{\leq s}$, we can describe $\omega_{1\sim N}$ in less than $N - 2\log N$ bits, by writing down every $\{\pi_y\}_{|y|\leq s}$ in verbatim, and then compressing $\{f_n\}_{n=1}^s$. (Note that $5\log N' > 2\log N$.) This contradicts the Kolmogorov-randomness of ω, thus Constraint 5.2 is satisfied.

Important problems. We define the following problems:

- Let $L^{\mathsf{hard}} = \{y : \exists x, f_{|x|}(x) = y\}$, then $L^{\mathsf{hard}} \in \mathsf{NP}$. (A valid witness for $y \in L^{\mathsf{hard}}$ would be a pair of strings (x, β) such that $\mathsf{Trapdoor}[x, y, \beta] \neq \perp$.)
- Let $\mathrm{GapApx}_{\delta(n)}\mathrm{MCSP}[a(n), b(n)]$ be the promise problem whose YES instances are truth tables with circuit complexity at most $a(n)$, and NO instances are truth tables that cannot be $\delta(n)$-approximated by size-$b(n)$ circuits.

In this section, we show that L^{hard} cannot be solved by circuits of size $2^{o(n)}$ with error 0.01 under the uniform distribution, and that L^{hard} Karp-reduces to MCSP. Actually, L^{hard} reduces to the weaker problem

$$\mathrm{GapApx}_{1/2+1/2^{n/100}}\mathrm{MCSP}[O(2^{n/2c}), \Omega(2^{n/100})].$$

Therefore, MCSP is 0.01-hard for $\mathsf{SIZE}[2^{o(N)}]$, under some samplable distribution.

Compressing a permutation by inverting it. Before we proceed, we need the following lemma, and the fact that this lemma relativizes:

Lemma 5.7 ([GGKT05])**.** *Let $\pi : \{0,1\}^t \to \{0,1\}^t$ be a permutation, A be a circuit that makes q queries to π, and inverts π on an ϵ fraction of inputs. Then we can describe π in*

$$2\log\binom{2^t}{a} + \log((2^t - a)!)$$

bits given A, where $a = \epsilon 2^t/(q+1)$.

5.3.1 Reducing L^{hard} to GapApxMCSP

The reduction is simple. Given a length-n input y to L^{hard}, consider the function $h^y : \{0,1\}^{\lceil c\log n\rceil} \times \{0,1\}^{\lceil c\log n\rceil} \to \{0,1\}$, where $h^y(\beta, r)$ is the inner product (over GF(2)) of $\pi_y^{-1}(\beta)$ and r. By iterating through $\Pi[y, \cdot]$, the truth table of h^y can be computed in $O(n^{2c})$ time.

If $y \in L^{\mathsf{hard}}$, then h^y has a circuit of size $O(n)$. Indeed, suppose $f(x) = y$, then $h^y(\beta, r)$ is the inner product of $\mathsf{Trapdoor}[x, y, \beta]$ and r. Therefore h^y has a circuit of size $O(n)$ that hardwires x.

On the other hand, if $y \notin L^{\mathsf{hard}}$, then we need to invert π_y in order to compute h^y, and such a circuit requires $\Omega(n^{c/200})$ size. We prove it formally in the following lemma.

Lemma 5.8. *Let n be an integer. For every $y \in \{0,1\}^n \setminus L^{\mathsf{hard}}$, any circuit C of size at most $n^{c/200}$ cannot $(1/2 + 1/n^{c/200})$-approximate h^y.*

Proof. Let $s = n^{c/10}$ and $\epsilon = 1/2n^{c/200}$. Suppose there is a circuit C of size $n^{c/200} \leq \frac{s}{(\lceil c \log n \rceil/\epsilon)^{10}}$ that $(1/2 + 2\epsilon)$-approximates h^y. By Corollary 2.15 on the permutation $\pi = \pi_y^{-1}$, there is a circuit C' of size s that computes π_y^{-1} successfully on an ϵ fraction of inputs.

Note that this circuit C' only has Π, $\mathsf{Trapdoor}$ or f_{PSPACE} gates of fanin at most $n^{c/200}$. Let $N = \ell_{\leq n^{c/200}}$, we show that the Kolmogorov complexity of $\omega_{1\sim N}$ (conditioned on N) is at most $N - n^{c/100} < N - 2\log N$, contradicting the choice of ω.

We start by storing everything in $\omega_{1\sim N}$ except the permutation π_y in verbatim, using at most $N - \log((2^t)!) + O(1)$ bits, where $t = \lceil c \log n \rceil$. We also store y which costs n bits, and C' which costs at most $n^{c/5}$ bits. After storing these data, we examine the oracle gates of C':

- Consider a query $\Pi[y', \alpha']$. If $y' \neq y$, then its answer is already stored, otherwise it can be simulated by one oracle call to an oracle for π_y.
- Any $\mathsf{Trapdoor}$ gate is completely determined by the already-stored value $\{\pi_{y'}\}_{y' \neq y}$ and $\{f_n\}$. This is because $y \notin L^{\mathsf{hard}}$, and $\mathsf{Trapdoor}$ gate does not depend on π_y at all.
- Any f_{PSPACE} gate is also completely determined by what we have already stored. This is because on input length $n^{c/200}$, by our padding to the f_{PSPACE} oracle, this oracle only depends on the oracle Π with input length at most $n^{1/100}$. Therefore π_y has no influence on f_{PSPACE} gates.

Now, π_y is a permutation over $\{0,1\}^t$ where $t = \lceil c \log n \rceil$, and a circuit C' making $q = n^{c/10}$ queries to π_y inverts π_y on an $\epsilon \geq 1/2n^{c/200}$ fraction of inputs. Let $a = \epsilon 2^t/(q+1)$, then $n^{4c/5} \leq a \leq n^c/2$. By Lemma 5.7, we can compress π_y in bit complexity

$$
\begin{aligned}
&2\log\binom{2^t}{a} + \log((2^t - a)!) \\
\leq\ & 2a(2 - \log(a/2^t)) + (2^t - a)(\log(2^t - a) - \log e) + \Theta(t) && (4)\\
\leq\ & \log((2^t)!) - a(\log(2^t - a) - 4 - \log e - 2\log(2^t/a)) + \Theta(t) && (5)\\
\leq\ & \log((2^t)!) - a(\log(n^c - n^c/2) - 4 - \log e - 2\log(2n^{c/5})) + \Theta(t) && (6)\\
\leq\ & \log((2^t)!) - \Theta(n^{4c/5}\log n). && (7)
\end{aligned}
$$

Here, Eq. (4) uses $\log\binom{n}{k} \leq k(2 - \log(k/n))$ (Claim 2.17) and $\log(n!) = n(\log n - \log e) + \Theta(\log n)$, Eq. (5) uses $\log((2^t)!) = 2^t(t - \log e) + \Theta(t)$, and Eq. (6) and (7) uses $t = \lceil c \log n \rceil$ and $n^{4c/5} \leq a \leq n^c/2$.

Therefore, we can describe $\omega_{1\sim N}$ in

$$N - \log((2^t)!) + 2n + n^{c/5} + \log((2^t)!) - \Theta(n^{4c/5}\log n) + O(1)$$

bits, which is less than $N - n^{c/100}$ bits, contradicting the Kolmogorov-randomness of ω. □

5.3.2 Hardness of L^{hard}

The hardness of L^{hard} is already shown in [Wee06], and the proof is also by compression method. We present a sketch here.

Lemma 5.9. *No circuit of size $2^{0.1n}$ can solve L^{hard} correctly on 0.99 fraction of inputs.*

Proof Sketch. Suppose there is a circuit C of size $2^{0.1n}$ that solves L^{hard} correctly on 0.99 fraction of inputs. Let $N = \ell_{\leq 2^{0.1n}}$, we compress $\omega_{1\sim N}$ in $N - 10 \cdot 2^{0.1n} \leq N - 2\log N$ bits.

First, we store everything in $\omega_{1\sim N}$ except f_n by verbatim, using $N - n2^n$ bits. Lemma 7 of [Wee06] showed how to represent the function f_n by $2^n(n - 0.35)$ bits, given our circuit C with size $2^{0.1n}$. Then we store the integer n and the circuit C using $2^{0.2n}$ bits. We can describe $\omega_{1\sim N}$ in

$$N - n2^n + 2^n(n - 0.35) + 2^{0.2n} < N - 10 \cdot 2^{0.1n} < N - 2\log N$$

bits, contradicting the Kolmogorov-randomness of ω. □

5.4 Pessiland III: MINKT Is Hard On Average

Our construction also works for other meta-computational problems, such as GapMINKT. In particular, let σ, τ be two polynomials, we define $\mathrm{Gap}_{\sigma,\tau}$MINKT as the following promise problem:

$$\mathrm{Gap}_{\sigma,\tau}\text{MINKT.YES} = \{(x, 1^s, 1^t) : \mathrm{K}^t(x) \le s\},$$
$$\mathrm{Gap}_{\sigma,\tau}\text{MINKT.NO} = \{(x, 1^s, 1^t) : \mathrm{K}^{\tau(t)}(x) > \sigma(s)\}.$$

Theorem 5.10. *Let c be any large constant. There is a relativized world where there is no auxiliary-input one-way functions, but under some samplable distribution, Gap_{s^c,t^c}MINKT cannot be 0.99-approximated by circuits of $2^{o(n)}$ size.*

Proof Sketch. In this world, apart from $\{f_n\}_{n\in\mathbb{N}}$, we also have a function π that maps every string of length n to a (random) string of length n^{c+1}. We have the following oracles:

- $\Pi\left[y, 1^{|y|^{(c+1)^2}}\right]$ outputs $\pi(y)$ given a padding of length $|y|^{(c+1)^2}$.
- $\mathsf{Trapdoor}[x, y]$ outputs $\pi(y)$ if $f_{|x|}(x) = y$, and outputs $\perp$ otherwise.
- $f_{\mathsf{PSPACE}}\left[M, 1^s, 1^{(|M|+s)^{(c+1)^2}}\right]$ outputs 1 if M is a Turing machine that does not invoke the $\mathsf{Trapdoor}$ oracle, and accepts the empty input within space complexity s; otherwise it outputs 0. We also require f_{PSPACE} to receive a padding of length $(|M| + s)^{(c+1)^2}$.

Again, the language $L^{\mathsf{hard}} = \{y : \exists x, f_{|x|}(x) = y\}$ is not 0.99-approximated by $2^{o(n)}$-size circuits, and $L^{\mathsf{hard}} \in \mathsf{NP}$. It is easy to see that L^{hard} reduces to GapMINKT: Given an input y of length n to L^{hard}, we can compute $\pi(y)$ in $O\left(n^{(c+1)^2}\right)$ time by the oracle Π.

- If $y \in L^{\mathsf{hard}}$ and $y = f(x)$, then $\pi(y) = \mathsf{Trapdoor}[x, y]$, so $\mathrm{K}^{O(n^{c+1})}(\pi(y)) \le 2n + O(1)$.
- If $y \notin L^{\mathsf{hard}}$, then let $t = 0.9n^{(c+1)^2}$, $\pi(y)$ is "independent" from every oracle with fanin at most t. Therefore, $\mathrm{K}^t(\pi(y)) \ge n^{c+1} - O(1)$.

It follows from the hardness of L^{hard} that Gap_{s^c,t^c}MINKT is hard on average in this oracle world. Again, by Lemma 5.4, there are no AIOWFs in this world. □

Remark 5.11. The Gap_{s^c,t^c}MINKT has much larger gap than the GapMINKT variants defined in [Hir18, Hir20a]. For example, to solve the GapMINKT in [Hir18], we need to approximate the shortest program within additive error $\tilde{O}(\sqrt{s})$, but here we only need to c-approximate $\log s$.

5.5 Proof of Lemma 5.3

Lemma 5.3. *Let s be an integer. For every circuit $C : \{0,1\}^n \to \{0,1\}^m$ of size s and every distribution $\mathcal{D}$ over $\{0,1\}^n$ that can be described in $O(\log n)$ bits, there is a circuit C' of $\mathrm{poly}(s)$ size without $\mathsf{Trapdoor}$ gates, such that*

$$\Pr_{\mathbf{z}\leftarrow\mathcal{D}}[C(\mathbf{z}) \ne C'(\mathbf{z})] \le 1/s.$$

In this section, we fix the distribution $\mathcal{D}$, and for two circuits C, C', we use $\Delta(C, C')$ as a shorthand for $\Pr_{\mathbf{z}\leftarrow\mathcal{D}}[C(\mathbf{z}) \ne C'(\mathbf{z})]$.

Our strategy is to eliminate every $\mathsf{Trapdoor}$ gate from bottom to top. For each $\mathsf{Trapdoor}$ gate, we will construct a small set H, and replace this gate by

$$\mathsf{Trapdoor}_H[x, y, \beta] = \begin{cases} \mathsf{Trapdoor}[x, y, \beta] & \text{if } f_{|x|}(x) = y \boxed{\text{ and } x \in H}, \\ \perp & \text{otherwise.} \end{cases}$$

Intuitively, H is the set of "heavy hitter" of this $\mathsf{Trapdoor}$ gate. The distribution $\mathcal{D}$ over the inputs to C induces a certain distribution over the inputs to the $\mathsf{Trapdoor}$ gate, and H can be seen as the set of strings x that appear frequently in this distribution. We will prove that since $\{f_n\}_{n\in\mathbb{N}}$ are "random" functions, if $x \notin H$ then $\mathsf{Trapdoor}[x, y, \beta]$ is likely to be $\perp$.

Therefore, $\mathsf{Trapdoor}_H$ will be a good approximation of $\mathsf{Trapdoor}$. Let C' be the circuit obtained from C by replacing each $\mathsf{Trapdoor}$ gate by a suitable $\mathsf{Trapdoor}_H$ gate, then C and C' are close. If for each $\mathsf{Trapdoor}$ gate, the set H we choose is small, then we can write C' as a small circuit without $\mathsf{Trapdoor}$ gates, and we are done.

Construction of C'. Let $C : \{0,1\}^n \to \{0,1\}^{n'}$ be a size-s circuit. Our goal is to find a circuit C' of size $\mathrm{poly}(s)$ such that $\Delta(C, C') \le 1/s$.

We set $\epsilon = 1/s^{10}$. Let $g_1, g_2, \ldots, g_s$ be the sequence of $\mathsf{Trapdoor}$ gates in C from bottom to top, i.e. for $i < j$, the sub-circuit with g_i as output gate does not contain g_j. We will define $H_1, H_2, \ldots, H_s$ in this order, each with size at most $\mathrm{poly}(s)$, and finally replace each g_i by a $\mathsf{Trapdoor}_{H_i}$ gate. For every $1 \le i \le s$, we define some intermediate circuits:

- C_i is the circuit obtained by replacing g_j with $\mathsf{Trapdoor}_{H_j}$ for each $1 \le j \le i$. (We also define $C_0 = C$.)
- C_i^{sub} is the sub-circuit of C_i with g_i (i.e. $\mathsf{Trapdoor}_{H_i}$) as output gate.
- $C_i^{\mathsf{sub}'}$ is the sub-circuit of C_{i-1} with g_i as output gate. (The only difference between C_i^{sub} and $C_i^{\mathsf{sub}'}$ is that g_i is replaced by a $\mathsf{Trapdoor}_{H_i}$ gate in C_i^{sub}, while g_i is a genuine $\mathsf{Trapdoor}$ gate in $C_i^{\mathsf{sub}'}$.)

Fix $1 \le i \le s$, the precise definition of H_i is as follows. Let g_i be a $\mathsf{Trapdoor}$ gate of fanin $2m_i + \lceil c \log m_i \rceil$. If $m_i \le 20 \log s$, then we let $H_i = H_{i-1} \cup \{0,1\}^{m_i}$. It is easy to see that $\mathsf{Trapdoor}_{H_i}$ coincides with $\mathsf{Trapdoor}$. If $m_i > 20 \log s$, we decompose the circuit $C_i^{\mathsf{sub}'}$ as

$$C_i^{\mathsf{sub}'}(z) = \mathsf{Trapdoor}[C_x(z), C_y(z), C_\beta(z)],$$

where $C_x : \{0,1\}^n \to \{0,1\}^{m_i}$, $C_y : \{0,1\}^n \to \{0,1\}^{m_i}$ and $C_\beta : \{0,1\}^n \to \{0,1\}^{\lceil c \log m_i \rceil}$ are size-s circuits. Note that each $\mathsf{Trapdoor}$ gate g_j in C_x, C_y, C_β is replaced by a $\mathsf{Trapdoor}_{H_j}$ gate. Now, for every string $x \in \{0,1\}^{m_i}$, we define α_x as the probability over a random $\mathbf{z} \leftarrow \mathcal{D}$ that $C_x(\mathbf{z}) = x$. We let

$$H_i = H_{i-1} \cup \{x \in \{0,1\}^{m_i} : \alpha_x \ge \epsilon^2\}.$$

Note that only strings of length m_i in H_i will take a role in defining $\mathsf{Trapdoor}_{H_i}$. However, for convenience (of defining further $H_{i'}$ $(i' > i)$), we also carry everything in H_{i-1} with H_i.

If $m_i \le 20 \log s$, then $|H_i| \le |H_{i-1}| + 2^{m_i} \le |H_{i-1}| + 1/\epsilon^2$. If $m_i > 20 \log s$, then as $\sum_x \alpha_x = 1$, it is also easy to see that $|H_i| \le |H_{i-1}| + 1/\epsilon^2$. Therefore, $|H_i| \le i/\epsilon^2$ for every i.

Finally, we define $C' = C_s$. Equivalently, C' is the circuit obtained by replacing every gate g_i by a $\mathsf{Trapdoor}_{H_i}$ gate. Note that C' is equivalent to a circuit of size $\mathrm{poly}(s)$ without $\mathsf{Trapdoor}$ gates. Since the portion of C_{i-1} and C_i outside $C_i^{\mathsf{sub}'}$ (or C_i^{sub}) are the same, we have $\Delta(C_{i-1}, C_i) \le \Delta(C_i^{\mathsf{sub}'}, C_i^{\mathsf{sub}})$. In what follows, we will prove that $\Delta(C_i^{\mathsf{sub}'}, C_i^{\mathsf{sub}}) \le 1/s^2$ for every integer $1 \le i \le s$, hence

$$\Delta(C, C') \le \sum_{i=1}^{s} \Delta(C_{i-1}, C_i) \le \sum_{i=1}^{s} \Delta(C_i^{\mathsf{sub}'}, C_i^{\mathsf{sub}}) \le 1/s.$$

Theorem 5.12. *Let $1 \le i \le s$, then $\Delta(C_i^{\mathsf{sub}'}, C_i^{\mathsf{sub}}) \le 1/s^2$.*

Proof. When $m_i \le 20 \log s$, we have $\Delta(C_i^{\mathsf{sub}'}, C_i^{\mathsf{sub}}) = 0$. Thus, in what follows we will assume that $m_i > 20 \log s$.

Warm-up: a proof assuming f_{m_i} is random. To get started, let us pretend that $f_{m_i} : \{0,1\}^{m_i} \to \{0,1\}^{m_i}$ is a random function, drawn from the following distribution. We assume that the set H_i and every function $\{f_{m'}\}_{m' \neq m_i}$ are fixed. We also assume that the values of f_{m_i} on inputs in H_i are also fixed. Denote $\Sigma = \{0,1\}^{m_i} \setminus H_i$, for every $x \in \Sigma$, $f_{m_i}(x)$ is independently drawn from $\{0,1\}^m$ uniformly at random.

Recall that

$$C_i^{\mathsf{sub}'}(z) = \mathsf{Trapdoor}[C_x(z), C_y(z), C_\beta(z)].$$

For every string $x \in \Sigma$, we introduce the random variables $\mathbf{p}_x$:

$$\mathbf{p}_x = \Pr_{\mathbf{z} \leftarrow \mathcal{D}}[C_x(\mathbf{z}) = x \text{ and } C_y(\mathbf{z}) = f_{m_i}(x)]. \tag{8}$$

(Here, $\mathbf{p}_x$ is a random variable over the choice of f_{m_i}, and for every fixed f_{m_i}, it is defined as a probability over the choice of $\mathbf{z}$.) Consider the random variable $\mathbf{P} = \sum_{x \in \Sigma} \mathbf{p}_x$, then $\Delta(C_i^{\mathsf{sub}'}, C_i^{\mathsf{sub}}) \leq \mathbf{P}$. This is because $C_i^{\mathsf{sub}'}(z) \neq C_i^{\mathsf{sub}}(z)$ only when $C_y(z) = f_{m_i}(C_x(z))$ and $C_x(z) \in \Sigma$.

Recall that $\alpha_x = \Pr_{\mathbf{z} \leftarrow \mathcal{D}}[C_x(\mathbf{z}) = x]$. Therefore, for every $x \in \Sigma$, α_x is an upper bound for $\mathbf{p}_x$. Note that each $\mathbf{p}_x$ only depends on $f_{m_i}(x)$, thus the random variables $\{\mathbf{p}_x\}_{x \in \Sigma}$ are independent. We can use Hoeffding's inequality (Theorem 2.16):

$$\Pr[\mathbf{P} - \mathbb{E}[\mathbf{P}] \geq t] \leq e^{-2t^2 / \sum_{x \in \Sigma} \alpha_x^2}. \tag{9}$$

It is easy to see that $\sum_{x \in \Sigma} \alpha_x \leq 1$ and $\max_{x \in \Sigma} \alpha_x \leq \epsilon^2$. Therefore $\sum_{x \in \Sigma} \alpha_x^2 \leq \epsilon^2$. We calculate $\mathbb{E}[\mathbf{P}]$:

$$\begin{aligned}
\mathbb{E}[\mathbf{P}] &= \Pr_{f_{m_i}; \mathbf{z} \leftarrow \mathcal{D}}[C_x(\mathbf{z}) \in \Sigma \text{ and } C_y(\mathbf{z}) = f_{m_i}(C_x(\mathbf{z}))] \\
&= \mathop{\mathbb{E}}_{\mathbf{z} \leftarrow \mathcal{D}}\left[\mathbb{I}[C_x(\mathbf{z}) \in \Sigma] \cdot \Pr_{f_{m_i}}[C_y(\mathbf{z}) = f_{m_i}(C_x(\mathbf{z}))]\right] \\
&\leq \frac{1}{2^{m_i}}.
\end{aligned}$$

Plugging $t = \sqrt{\epsilon}$ and $\mathbb{E}[\mathbf{P}] \leq 1/2^{m_i}$ into Eq. (9), we have that over the choice of f_{m_i},

$$\Pr[\mathbf{P} \geq \sqrt{\epsilon} + 1/2^{m_i}] \leq e^{-2t^2/\epsilon^2} \leq e^{-2/\epsilon}. \tag{10}$$

Since $\mathbf{P} \geq \Delta(C_i^{\mathsf{sub}'}, C_i^{\mathsf{sub}})$ and $\sqrt{\epsilon} + 1/2^{m_i} \leq 1/s^2$, we also have

$$\Pr[\Delta(C_i^{\mathsf{sub}'}, C_i^{\mathsf{sub}}) \geq 1/s^2] \leq e^{-2/\epsilon}. \tag{11}$$

The incompressibility argument. Recall that f_{m_i} is not a random variable, but a fixed incompressible function. Now, assuming $P = \sum_{x \in \Sigma} p_x \geq 1/s^2$, we use the above argument to compress the functions $\{f_n\}_{n=1}^s$, contradicting Constraint 5.2. Let $N = \sum_{n=1}^s n2^n$, then $N \leq 3^s$. We show that if $\Delta(C_i^{\mathsf{sub}'}, C_i^{\mathsf{sub}}) \geq 1/s^2$, then conditioned on the portion of Π and f_{PSPACE} with input length at most s, we can compress $\{f_n\}_{n=1}^s$ into at most $N - 10s < N - 5 \log N$ bits.

First, we write down $\mathcal{D}$ which costs $O(\log n)$ bits. Then, we write down the circuit C and subsets $H_1, H_2, \ldots, H_i$. For every $x \in H_i$, we also write down $f_{|x|}(x)$. Note that we do not have enough space to write them down in verbatim, but fortunately we do not have to. For every $1 \leq j \leq i$, given C, H_{j-1}, and $\{f_{|x|}(x)\}_{x \in H_{j-1}}$, we can compute H_j as follows:

- Compute $C_j^{\mathsf{sub}'}$ from the description of C, and decompose $C_j^{\mathsf{sub}'} = \mathsf{Trapdoor}[C_x(z), C_y(z), C_\beta(z)]$.
- Replace each $\mathsf{Trapdoor}$ gate in C_x, C_y or C_β by the corresponding $\mathsf{Trapdoor}_{H_k}$ gate; this is possible since $k < j$ and we already know H_k and $\{f_{m_k}(x)\}_{x \in H_k}$.

- Compute the set $H_j = H_{j-1} \cup \{x : \Pr_{\mathbf{z} \leftarrow \mathcal{D}}[C_x(\mathbf{z}) = x] \geq \epsilon^2\}$.

Then, we use $|H_j \setminus H_{j-1}| \cdot m_j$ bits to write down $f_{m_j}(x)$ for every $x \in H_j \setminus H_{j-1}$, in some canonical order. It follows that we only need $|C| + \sum_{x \in H_i} |x|$ bits to record C, H_i, and $\{f_{|x|}(x)\}_{x \in H_i}$.

Next we write down the functions $\{f_l\}_{l \neq m_i}$. Note that we only need to write $f_{|x|}(x)$ for the strings x that are not in H_i. This costs

$$\sum_{1 \leq l \leq s, l \neq m_i} (2^l - |\{0,1\}^l \setminus H_i|) \cdot l$$

bits. The total number of bits we have used so far is

$$\begin{aligned} &O(\log n) + |C| + \sum_{x \in H_i} |x| + \sum_{1 \leq l \leq s,\, l \neq m_i} (2^l - |\{0,1\}^l \setminus H_i|) \cdot l \\ =\,&O(\log n) + N + |C| - (2^{m_i} - |\{0,1\}^{m_i} \cap H_i|) \cdot m_i. \end{aligned}$$

Now, let $\mathcal{F}$ be the set of functions $f : \{0,1\}^{m_i} \to \{0,1\}^{m_i}$ such that f is consistent with f_{m_i} on H_i, and if we replace f_{m_i} by f, then $P \geq 1/s^2$. From Eq. (10), we can see that $|\mathcal{F}| \leq 2^{m_i \cdot \sigma} e^{-2/\epsilon}$, where $\sigma = |\{0,1\}^{m_i} \setminus H_i|$. For every function f, given the information we have already recorded, there is an algorithm that determines whether $f \in \mathcal{F}$ in finitely many steps: Given H_i and $\{f_{|x|}(x)\}_{x \in H_i}$, we can compute the descriptions of C_x, C_y, and compute each p_x from Eq. (8). Note that f_{m_i} is replaced by f here. Then we can compute $P = \sum_x p_x$ and compare it with $1/s^2$ to decide whether $f \in \mathcal{F}$.

Therefore, it suffices to record a number k such that f_m is the lexicographically k-th smallest function in $\mathcal{F}$, and this number costs $m_i \cdot \sigma - (2/\epsilon) \log e + O(1)$ bits. It follows that we can record $\{f_n\}_{n=1}^s$ in

$$\begin{aligned} &O(\log n) + N + |C| + m_i \cdot \sigma - (2/\epsilon) \log e - 2^{m_i} \cdot m_i + |\{0,1\}^{m_i} \cap H_i| \cdot m_i + O(1) \\ \leq\,&N + O(s \log s) - (2/\epsilon) \log e \\ \leq\,&N - 10s \end{aligned}$$

bits, contradicting Constraint 5.2. Thus we have $\Delta(C_i^{\mathsf{sub}'}, C_i^{\mathsf{sub}}) \leq P < 1/s^2$. □

5.6 Proof of Lemma 5.4

Lemma 5.4. *There are no auxiliary-input one-way functions in our oracle worlds.*

Proof. It suffices to show an algorithm that inverts every circuit in the sense of Claim 2.10. Let $C, C' : \{0,1\}^p \to \{0,1\}^q$ be two circuits, in this section, we will denote $\Delta(C, C') = \Pr_{\mathbf{z} \leftarrow \mathcal{U}_p}[C(\mathbf{z}) \neq C'(\mathbf{z})]$. That is, instead of the (arbitrary) distribution $\mathcal{D}$ as in Section 5.5, here we define $\Delta(C, C')$ to be the distance between C and C' w.r.t. the *uniform* distribution.

Let $C : \{0,1\}^p \to \{0,1\}^q$ be a circuit of size s, then by Lemma 5.3, there is a circuit C' of size $s' = \mathrm{poly}(s)$ without any $\mathsf{Trapdoor}$ gate, such that $\Delta(C, C') \leq 1/s$. Unfortunately, the circuit C' may not be efficiently computable from C. (It seems that we need to be able to compute each f_{m_i} in order to compute C' from C.) However, we can still use the following sampling trick to "learn" C'.

We sample $\ell = s's^3$ random inputs $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\ell \leftarrow \mathcal{U}_p$. Then for every i, we define $\mathbf{y}_i = C(\mathbf{x}_i)$. We find a circuit $\widetilde{C}$ of size s' that does not use $\mathsf{Trapdoor}$ gates, such that the "empirical error"

$$\mathsf{err}(\widetilde{C}) := \Pr_{i \leftarrow [\ell]}[\widetilde{C}(\mathbf{x}_i) \neq \mathbf{y}_i]$$

is minimized. Given the data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i\in[\ell]}$, we can enumerate $\widetilde{C}$ and calculate $\mathsf{err}(\widetilde{C})$ in $\mathrm{poly}(s')$ space without invoking $\mathsf{Trapdoor}$ gates, therefore we can also compute $\widetilde{C}$ in $\mathrm{poly}(s')$ time with the aid of the f_{PSPACE} oracle. We claim that $\Delta(C, \widetilde{C}) \le 3/s$ with high probability.

First, fix an arbitrary circuit C_{far} such that $\Delta(C, C_{\mathsf{far}}) > 3/s$. Let $\boldsymbol{\delta}_i \in \{0, 1\}$ be the indicator of whether $C_{\mathsf{far}}(\mathbf{x}_i) = \mathbf{y}_i$, and $\boldsymbol{\delta} = \sum_{i=1}^{\ell} \boldsymbol{\delta}_i$, then $\mathbb{E}[\boldsymbol{\delta}] > 3\ell/s$, where the expectation is over the random choices of $\{\mathbf{x}_i\}_{i\in[\ell]}$. By Theorem 2.16, we have

$$\Pr[\mathsf{err}(C_{\mathsf{far}}) \le 2/s] = \Pr[\mathbb{E}[\boldsymbol{\delta}] - \boldsymbol{\delta} > \ell/s] \le \exp(-2\ell/s^2).$$

By a union bound over all circuits of size at most s', the probability that there is a circuit C_{far} of size at most s' such that $\Delta(C, C_{\mathsf{far}}) > 3/s$ but $\mathsf{err}(C_{\mathsf{far}}) \le 2/s$ is at most

$$\exp(O(s' \log s') - 2\ell/s^2) \le \exp(-s's).$$

On the other hand, consider the circuit C' which is $(1/s)$-close to C. Again, by Theorem 2.16, the probability that $\mathsf{err}(C') > 2/s$ is at most $\exp(-2\ell/s') \le \exp(-s's)$. Therefore, w.p. at least $1 - 2e^{-s's}$, we have $\Delta(C, \widetilde{C}) \le 3/s$.

Things become easy when we find a circuit $\widetilde{C}$ such that $\Delta(C, \widetilde{C}) \le 3/s$ and $\widetilde{C}$ contains no $\mathsf{Trapdoor}$ gate. Given an output $\mathbf{y} \leftarrow C(\mathcal{U}_p)$, we find a uniformly random preimage $\widetilde{\mathbf{z}}$ over the set $\widetilde{C}^{-1}(\mathbf{y})$. We output "failure" if $\widetilde{C}^{-1}(\mathbf{y}) = \varnothing$. This preimage can be found in $\mathrm{poly}(s)$ space with s random bits, and the distribution of $\widetilde{\mathbf{z}}$ is $1/2^{O(s)}$-close to the uniform distribution over $\widetilde{C}^{-1}(\mathbf{y})$. Therefore, we can use an f_{PSPACE} oracle (together with s random bits) to find a random $\widetilde{\mathbf{z}}$ sampled uniformly (with $1/2^{O(s)}$ bias) from $\widetilde{C}^{-1}(\mathbf{y})$, in $\mathrm{poly}(s)$ time. Then we output $\widetilde{\mathbf{z}}$ if $C(\widetilde{\mathbf{z}}) = \mathbf{y}$, and output "failure" otherwise.

We prove that the above algorithm inverts $\mathbf{y}$ with probability $1 - O(1/s)$. Let $\mathcal{D}$ be the distribution of $C(\mathcal{U}_p)$, and $\widetilde{\mathcal{D}}$ be the distribution of $\widetilde{C}(\mathcal{U}_p)$. Since $\Delta(C, \widetilde{C}) \le 3/s$, the statistical distance between $\mathcal{D}$ and $\widetilde{\mathcal{D}}$ is also at most $3/s$. If the algorithm is given $\mathbf{y} \leftarrow \widetilde{\mathcal{D}}$ instead of $\mathbf{y} \leftarrow \mathcal{D}$, then the distribution of $\mathbf{z}$ is $1/2^{O(s)}$-close to the uniform distribution, and the probability that the algorithm outputs "failure" is at most the probability that $C(\mathbf{z}) \neq \widetilde{C}(\mathbf{z})$, which is at most $3/s + 1/2^{O(s)} < 4/s$. Therefore, if the algorithm is given $\mathbf{y} \leftarrow \mathcal{D}$, then its failure probability is at most $7/s$. □

6 Limits of "Robust" Reductions to GapMINKT Oracles

We study the limitation of *robust* reductions to the promise problem GapMINKT in relativized worlds. Here, a reduction is *robust*, if it is correct even if the oracle for GapMINKT knows the input (and possibly nondeterministic bits) of the reduction, and answers adversarially on every query not in the promise of GapMINKT.

Definition 6.1 (Robust Reductions)**.** Let $L = (L.\textsc{Yes}, L.\textsc{No})$ be a promise problem, and L' be a computational task (e.g. a language, a promise problem, or solving a problem on average-case).

- We say that a coNP-Turing reduction M from L' to L is *robust*, if M is a co-nondeterministic machine such that the following holds. For any adversary that (*knows the input and the nondeterministic bits of* M and) answers L-queries asked by M, as long as the adversary answers correctly on $L.\textsc{Yes} \cup L.\textsc{No}$ (and arbitrarily otherwise), M solves L' successfully.
- Similarly, we say that a $\mathsf{P}_{/\mathrm{poly}}$-Turing reduction C from L' to L is *robust*, if C is a circuit such that the following holds. For any adversary that (*knows the input of* C and) answers L-queries asked by C, as long as the adversary answers correctly on $L.\textsc{Yes} \cup L.\textsc{No}$, C solves L' successfully.

In this section, we show that techniques that are both robust and relativizing cannot prove either of the following statements:

1. GapMINKT is NP-hard under coNP-Turing reductions;[8] or
2. GapMINKT is the hardest problem in DistNP.

Definition of GapMINKT**.** Fix a large enough constant c. We define GapMINKT as the following promise problem:

$$\text{GapMINKT}^{\Pi}.\text{YES} = \{(x, 1^s, 1^t) : \text{K}^{t,\Pi}(x) \leq s\},$$
$$\text{GapMINKT}^{\Pi}.\text{NO} = \{(x, 1^s, 1^t) : \text{K}^{ct,\Pi}(x) > s + c \log t\}.$$

We remark that the "gap" of GapMINKT in our definition is quite small, compared to the variants of GapMINKT in [Hir18, Hir20b, Hir20a]. Therefore, our GapMINKT is harder to solve, and our negative results are stronger.

Weakness of GapMINKT**.** As discussed in Section 1.3.4, the weakness of GapMINKT is that for two oracles worlds $\mathcal{O}$ and $\mathcal{O}'$ that are very close, it does not provide information about which world we are in. We summarize it in the following lemma:

Lemma 6.2. *Let c be a large enough constant, $\mathcal{O}$ and $\mathcal{O}'$ be two oracles that only differ at one input. Let x be a string, t be an integer, $s = \text{K}^{t,\mathcal{O}}(x)$. Then $\text{K}^{\sqrt{c}\cdot t,\mathcal{O}'}(x) \leq s + (c/3)\log t$.*

Proof. Let M be the $\mathcal{O}$-oracle Turing machine with description length s that outputs x in t steps. Let α be the input at which $\mathcal{O}$ and $\mathcal{O}'$ differs. If M does not query $\mathcal{O}[\alpha]$, then it is clear that $\text{K}^{t,\mathcal{O}'}(x) \leq s$. Otherwise, let i be the timestamp of the first query $\mathcal{O}[\alpha]$ that M makes. Consider the Turing machine M' with oracle access to $\mathcal{O}'$, that hardcodes i, simulates M, records the query α at time i and then flips every response of the query $\mathcal{O}'[\alpha]$. Its behavior is exactly the same as M, so it also outputs x. For a large enough constant c, the description length of M' is at most $s + (c/3)\log t$, and M' runs in at most $\sqrt{c}\cdot t$ steps. Therefore $\text{K}^{\sqrt{c}\cdot t,\mathcal{O}'}(x) \leq s + (c/3)\log t$. □

6.1 NP-Intermediateness of GapMINKT under coNP-Turing Reductions

In this section, we construct a relativized world where GapMINKT $\notin$ coNP, but coNP-Turing reductions that are robust cannot show NP-hardness of GapMINKT either.

Theorem 6.3. *There is a relativized world such that the following are true:*

*(*GapMINKT *Is Not* NP*-Complete) There is a language $L^{\mathsf{hard}} \in \mathsf{NP}$, such that there is no co-nondeterministic Turing reduction from L^{hard} to* GapMINKT *that runs in $2^N/N^{\omega(1)}$ time.*

*(*GapMINKT *Is Not Easy) Every co-nondeterministic Turing machine that runs in $2^N/N^{\omega(1)}$ time does not solve* GapMINKT*.*

Proof. Our oracle world has an oracle $\mathcal{O}$ that diagonalizes against every nondeterministic Turing machine. The hard problem in NP is simply

$$L^{\mathsf{hard}} = \{0^n : \mathcal{O} \cap \{0,1\}^n \neq \varnothing\}.$$

Let $\{M_i\}_{i\in\mathbb{N}}$ be an enumeration of nondeterministic Turing machines with time complexity $2^N/N^{\omega(1)}$. We proceed in stages, where in each stage, we will fix finitely many inputs to $\mathcal{O}$. Let N_i be the smallest integer such that at the end of stage i, every fixed string has length at most N_i. In stage $2i$, we will guarantee that M_i with a GapMINKT oracle does not compute $\overline{L^{\mathsf{hard}}}$ (the complement of L^{hard}) in $2^N/N^{\omega(1)}$ time. In stage $2i+1$, we will guarantee that M_i does not compute the complement of GapMINKT in $2^N/N^{\omega(1)}$ time.

[8] Actually, for this result, it is possible to weaken the requirements of robust reductions; however for the other result, it is unclear to the authors how to weaken the requirement that the reduction has to be robust.

Stage $2i$. In this stage, we want that M_i does not compute $\overline{L^{\mathsf{hard}}}$ even given a GapMINKT oracle.

Let $N = N_{2i-1} + 1$, we simulate M_i on input 0^N. Let $\mathcal{O}_{2i-1}$ denote the oracle $\mathcal{O}$ at the end of stage $2i-1$. (That is, except for the inputs that we already fixed in the first $2i-1$ stages, $\mathcal{O}_{2i-1}$ rejects every input.) We will provide M_i the following oracle MINKT' as a $\mathrm{GapMINKT}^{\mathcal{O}}$ oracle:

$$\mathrm{MINKT}'[x, 1^s, 1^t] = \mathrm{MINKT}^{\mathcal{O}_{2i-1}}[x, 1^{s+(c/2)\log t}, 1^{\sqrt{c}\cdot t}].$$

Also, each time M_i asks a query to $\mathcal{O}$, if it is not already fixed, then we return 0 to this query. (Equivalently, we use $\mathcal{O}_{2i-1}$ to reply queries to $\mathcal{O}$.)

Recall that M_i is a nondeterministic Turing machine, and we want that M_i does not compute $\overline{L^{\mathsf{hard}}}$. Therefore, if some nondeterministic branch of M_i returns 1, we should put some strings into the length-N slice of $\mathcal{O}$, so that $0^N \notin \overline{L^{\mathsf{hard}}}$; if every nondeterministic branch of M_i returns 0, we should put nothing in the length-N slice of $\mathcal{O}$, so that $0^N \in \overline{L^{\mathsf{hard}}}$.

Case 1: Suppose that some nondeterministic branch of M_i returns 1. In this branch, M_i asks at most $2^N/N^{\omega(1)}$ queries to $\mathcal{O}$. We can simply find any string x of length N such that $\mathcal{O}[x]$ was not asked in this branch of M_i, and set $\mathcal{O}[x] = 1$. Since $\mathcal{O}$ and $\mathcal{O}_{2i-1}$ only differ at one input, we can see that MINKT' satisfies the promise of $\mathrm{GapMINKT}^{\mathcal{O}}$. Actually, let $(x, 1^s, 1^t)$ be an input:

- If $\mathrm{K}^{t,\mathcal{O}}(x) \le s$, then by Lemma 6.2, $\mathrm{K}^{\sqrt{c}\cdot t,\mathcal{O}_{2i-1}}(x) \le s + (c/3)\log t$, thus $(x, 1^s, 1^t) \in \mathrm{MINKT}'$.
- If $\mathrm{K}^{ct,\mathcal{O}}(x) > s + c\log t$, then by the contrapositive of Lemma 6.2, $\mathrm{K}^{\sqrt{c}\cdot t,\mathcal{O}_{2i-1}}(x) > s + (c/3)\log t$, thus $(x, 1^s, 1^t) \notin \mathrm{MINKT}'$.

Therefore, it is valid to replace the $\mathrm{GapMINKT}^{\mathcal{O}}$ oracle by MINKT'. For a suitable choice of $\mathrm{GapMINKT}^{\mathcal{O}}$ oracle, M_i accepts the input 0^N on some nondeterministic branch, thus does not solve $\overline{L^{\mathsf{hard}}}$ on input length N.

Case 2: Suppose that every nondeterministic branch of M_i returns 0. We simply let the length-N slice of $\mathcal{O}$ be empty. Since $\mathcal{O} = \mathcal{O}_{2i-1}$, MINKT' satisfies the promise of $\mathrm{GapMINKT}^{\mathcal{O}}$. Again, for a suitable choice of $\mathrm{GapMINKT}^{\mathcal{O}}$ oracle, M_i rejects the input 0^N on every nondeterministic branch, thus does not solve $\overline{L^{\mathsf{hard}}}$ on input length N.

Finishing stage $2i$. Note that if $\mathcal{O}$ contains strings of length $> N$, then MINKT' may not be consistent with the final $\mathrm{GapMINKT}^{\mathcal{O}}$ oracle. Therefore, we need to set $N_{2i} = 2^N$. For every string x with length at most N_{2i}, if $\mathcal{O}[x]$ is not already fixed, then we fix $\mathcal{O}[x] = 0$. Now, for every $(x, 1^s, 1^t)$ such that $t \le N_{2i}$ (which includes every GapMINKT query that M_i might ask), $\mathrm{MINKT}'[x, 1^s, 1^t]$ is indeed consistent with $\mathrm{GapMINKT}^{\mathcal{O}}[x, 1^s, 1^t]$.

Stage $2i+1$. In this stage, we want that M_i does not compute the complement of GapMINKT.

Let $N = N_{2i} + 2c\log N_{2i}$, $s = N_{2i} + O(1)$, and $t = 10cN$. Let x_{hard} be an input of length N such that

$$\mathrm{K}^{ct,\mathcal{O}_{2i}}(x_{\mathsf{hard}}) > s + c\log t,$$

where, again, $\mathcal{O}_{2i}$ is the oracle $\mathcal{O}$ at the end of stage $2i$. By Claim 2.12, such an input x_{hard} exists. Here, the choice of s and t ensures that if we "embed" x_{hard} in the length-$(N_{2i}+1)$ slice of $\mathcal{O}$, then there is a length-s program that outputs x_{hard} in time t.

We simulate M_i on input $(x_{\mathsf{hard}}, 1^s, 1^t)$. Each time M_i asks a query to $\mathcal{O}$, if the query is not fixed, we simply return 0. Now we want that M_i fails to solve the complement of GapMINKT, and there are two cases.

Case 1: Suppose some nondeterministic branch of M_i returns 1, then we want $\mathrm{K}^{t,\mathcal{O}}(x_{\mathsf{hard}}) \leq s$. On this branch, M_i asks at most $2^N/N^{\omega(1)} \ll 2^{N_{2i}-\lceil \log N \rceil}$ queries to $\mathcal{O}$. Therefore we can find a string pfx of length $N_{2i} - \lceil \log N \rceil$, such that pfx is not a prefix of any query that M_i made to $\mathcal{O}$. Then we "embed" x_{hard} into the length-$(N_{2i}+1)$ slice of $\mathcal{O}$:

$$\mathcal{O}[\mathsf{pfx} \circ j] = (x_{\mathsf{hard}})_j, \forall 0 \leq j < N.$$

Here, we regard j as a length-$\lceil \log N \rceil$ string, and $\mathsf{pfx} \circ j$ is the concatenation of pfx and j. Then we can see that M_i accepts the input $(x, 1^s, 1^t)$, but $\mathrm{K}^{t,\mathcal{O}}(x_{\mathsf{hard}}) \leq s$.

Case 2: Suppose every nondeterministic branch of M_i returns 0, then we want $\mathrm{K}^{ct,\mathcal{O}}(x_{\mathsf{hard}}) > s + c \log t$. In this case, for every string x of length at most ct, if $\mathcal{O}[x]$ is not fixed yet, then we fix $\mathcal{O}[x] = 0$. Now $\mathcal{O}$ and $\mathcal{O}_{2i}$ are the same oracle on input lengths up to ct. We have that M_i rejects the input $(x, 1^s, 1^t)$, but $\mathrm{K}^{ct,\mathcal{O}}(x_{\mathsf{hard}}) > s + c \log t$.

Finishing stage $2i+1$**.** Let N_{2i+1} be the length of the longest string currently fixed to $\mathcal{O}$. We arbitrarily fix every input of length at most N_{2i+1} to $\mathcal{O}$ that are not fixed yet, and finish stage $2i+1$. □

6.2 Is GapMINKT the Hardest Problem in DistNP?

In this section, we construct a relativized world where there is no robust reduction proving the statement "if GapMINKT is easy, then $\mathsf{DistNP} \subseteq \mathsf{AvgP}_{/\mathrm{poly}}$". In particular, we show that any circuit with GapMINKT gates that robustly solves some NP problem on a $(1/2+1/2^{o(n)})$-fraction of inputs requires $2^{\Omega(n)}$ size.

We recall that an oracle circuit C is a robust reduction from a DistNP problem $(L, \mathcal{D})$ to GapMINKT, if for every adversary $\mathcal{A}$ that (knows the input $\mathbf{x}$ to C and) answers GapMINKT queries, we have

$$\Pr_{\mathbf{x} \sim \mathcal{D}_n}[C^{\mathcal{A}}(\mathbf{x}) = L(\mathbf{x})] \geq 2/3.$$

Actually, we may assume that on a particular input, the circuit never asks the same query twice. This can be achieved by storing the answers for every previously asked query in a hash table. Then, there is an equivalent definition of robust reductions. We say that an oracle circuit C is a robust reduction from $(L, \mathcal{D}) \in \mathsf{DistNP}$ to GapMINKT, if with probability at least $2/3$ over a random input $\mathbf{x} \leftarrow \mathcal{D}_n$, for every oracle $\mathcal{O}$ consistent with GapMINKT, we have $C^{\mathcal{O}}(\mathbf{x}) = L(\mathbf{x})$. (Note that the weaker, and more natural, definition of Turing reductions to a promise problem switches two quantifiers: For every oracle $\mathcal{O}$ consistent with GapMINKT, $C^{\mathcal{O}}$ solves L on average.)

Theorem 6.4. *There is a relativized world such that the following holds. There is an* NP *problem* L^{hard}*, such that every circuit* C *that is a robust reduction from* $(L^{\mathsf{hard}}, \{\mathcal{U}_n\}_{n\in\mathbb{N}})$ *(i.e. a distributional problem which is* L^{hard} *paired with the uniform distribution) to* GapMINKT *requires size* $2^{\Omega(n)}$*.*

Proof. We consider the simplest world with a hard-on-average problem. For every integer n, we have a random permutation $\pi^n : \{0,1\}^n \to \{0,1\}^n$. We provide an oracle Π that on input strings α, β of the same length, outputs 1 if $\pi^{|\alpha|}(\alpha) = \beta$, and outputs 0 otherwise. At the end of this section, we will use the incompressibility method to rigorously define this world.

Recall that $\langle \cdot, \cdot \rangle$ denotes the inner product function over GF(2). Our hard language is

$$L^{\mathsf{hard}} = \{(\alpha, r) : \langle \pi^{|\alpha|}(\alpha), r \rangle = 0\}.$$

It is easy to see that $L^{\mathsf{hard}} \in \mathsf{NP}^{\Pi}$. (Actually, $(\mathsf{UP} \cap \mathsf{coUP})^{\Pi}$.) We will show that any circuit with GapMINKT gates that solves L^{hard} on a $(1/2+1/2^{0.005n})$-fraction of inputs requires $2^{0.005n}$

size. By Corollary 2.15, it suffices to show that any circuit with (Π and) GapMINKT gates that computes π^n on a $2^{-0.1n}$ fraction of inputs requires $2^{0.1n}$ size.

Our proof strategy will be similar as [GGKT05, Theorem 2.1]. Suppose there is a circuit C of size $2^{0.1n}$ with Π and GapMINKT gates that computes π^n (on a $2^{-0.1n}$ fraction of inputs). We show that given C and every other permutation $\{\pi^{n_0}\}_{n_0 \neq n}$, we can compress π^n into less than $\log_2((2^n)!)$ bits.

Compression algorithm. Without loss of generality, we assume that on input α, our circuit C always queries $\Pi[\alpha, C(\alpha)]$ at the end. We record C which has at most $2^{0.1n}$ gates. Besides C, we also record the following data to represent π^n.

- We write down a subset $\mathcal{S} \subseteq \{0,1\}^n$ which is "not too large", and the value of π^n on $\mathcal{S}$. Equivalently, we write down a list of pairs $(\alpha, \pi^n(\alpha))$ for every $\alpha \in \mathcal{S}$ (of course, in an information-theoretically optimal way).
- For every input $\alpha \in \{0,1\}^n \setminus \mathcal{S}$, we also record an integer p_α which is between 1 and $|C|$. This information will be helpful for recovering $\pi^n(\alpha)$.

Initially we let $\mathcal{S}$ be the set of strings $\alpha \in \{0,1\}^n$ on which C fails, i.e. for some oracle B consistent with $\mathrm{GapMINKT}^\Pi$, $\pi^n(\alpha) \neq C^{B,\Pi}(\alpha)$.

Let $\alpha_1, \alpha_2, \ldots, \alpha_{2^n}$ be the list of length-n strings in lexicographical order. We use the notation $\alpha \prec \alpha'$ to denote that α is lexicographically smaller than α'. We proceed in iterations. For each $1 \leq i \leq 2^n$, in the i-th iteration, if $\alpha_i \in \mathcal{S}$, we do nothing. If $\alpha_i \notin \mathcal{S}$, we will simulate the circuit C on input α_i, where $\mathrm{GapMINKT}^\Pi$ is replaced by a suitable oracle (consistent with the promise of $\mathrm{GapMINKT}^\Pi$) that *depends* on α_i. Also, we will potentially add some elements (but never α_i itself) into $\mathcal{S}$. At last we will record an integer p_{α_i}. In this iteration, we may assume that for every $\alpha \in \mathcal{S}$, $\pi^n(\alpha)$ is "fixed" (for the decompression algorithm). As we proceed in the lexicographical order, we can also assume that $\pi^n(\alpha)$ is "fixed" for every $\alpha \prec \alpha_i$.

In the i-th iteration, we will use the oracle "MINKT_{-i}" to replace $\mathrm{GapMINKT}^\Pi$, defined as follows. Let Π_{-i} be an oracle identical to Π, except that $\Pi_{-i}[\alpha_i, \pi^n(\alpha_i)] = 0$. On input $(x, 1^s, 1^t)$, the oracle MINKT_{-i} outputs 1 if $\mathrm{K}^{\sqrt{c}t,\Pi_{-i}}(x) \leq s + (c/3)\log t$, and outputs 0 otherwise. Since Π and Π_{-i} only differ at one input, we can show that MINKT_{-i} satisfies the promise of $\mathrm{GapMINKT}^\Pi$:

- if $\mathrm{K}^{t,\Pi}(x) \leq s$, then by Lemma 6.2, $\mathrm{K}^{\sqrt{c}t,\Pi_{-i}}(x) \leq s + (c/3)\log t$, thus $(x, 1^s, 1^t) \in \mathrm{MINKT}_{-i}$;
- if $\mathrm{K}^{ct,\Pi}(x) > s + c\log t$, then by the contrapositive of Lemma 6.2, $\mathrm{K}^{\sqrt{c}t,\Pi_{-i}}(x) > s + (c/3)\log t$, thus $(x, 1^s, 1^t) \notin \mathrm{MINKT}_{-i}$.

Now we define exactly what the i-th iteration does (if $\alpha_i \notin \mathcal{S}$). We run C on input α_i. Each time we make a query to Π, namely $\Pi[\alpha, \beta]$:

- We reply with $\Pi[\alpha, \beta]$ honestly.
- If $\alpha \succ \alpha_i$ and $\alpha \notin \mathcal{S}$, then we add α into $\mathcal{S}$.
- If $\alpha = \alpha_i$ and $\beta = \pi^n(\alpha_i)$, we write down that $p_{\alpha_i} = k$, where the current query is the k-th gate in C. Then we stop simulating C, finish the i-th iteration immediately, and proceed to the next iteration. (Since we require C to query its own output, i.e. $\Pi[\alpha_i, C(\alpha_i)]$, at the end, p_{α_i} will always be defined.)
- Otherwise we do not perform extra operations.

Each time we make a query to $\mathrm{GapMINKT}^\Pi$, namely $\mathrm{GapMINKT}^\Pi[x, 1^s, 1^t]$:

- We return the answer $\mathrm{MINKT}_{-i}[x, 1^s, 1^t]$. As discussed above, this is a valid answer for $\mathrm{GapMINKT}^{\Pi}[x, 1^s, 1^t]$.
- Suppose the returned answer is 1. Let M be any Π_{-i}-oracle machine of description length $s+(c/3)\log t$ that outputs x in $\sqrt{c}t$ time. For every query $\Pi'[\alpha, \beta]$ that M makes, if $\alpha \succ \alpha_i$ and $\alpha \notin \mathcal{S}$, then we add α into $\mathcal{S}$.

After the 2^n-th iteration terminates, we obtain a set $\mathcal{S} \subseteq \{0,1\}^n$ and an integer p_α for every $\alpha \in \{0,1\}^n \setminus \mathcal{S}$.

Decompression algorithm. We need to show that the above data suffices to recover π^n. As we stored $\pi^n(\alpha)$ for every $\alpha \in \mathcal{S}$ explicitly, we only need to recover $\pi^n(\alpha)$ for every $\alpha \notin \mathcal{S}$.

Again, let $\alpha_1, \alpha_2, \ldots, \alpha_{2^n}$ be the list of length-n strings in lexicographical order. For each $1 \le i \le 2^n$, if $\alpha_i \notin \mathcal{S}$, then we need to recover $\pi^n(\alpha_i)$. We simulate C on the input α_i.

Whenever C makes a query $\Pi[\alpha, \beta]$, we know that either $\alpha \preceq \alpha_i$ or $\alpha \in \mathcal{S}$. If $\alpha \prec \alpha_i$ or $\alpha \in \mathcal{S}$, then we already know $\pi^n(\alpha)$, thus also know which value we should reply to $\Pi[\alpha, \beta]$. Otherwise (i.e. $\alpha = \alpha_i$), suppose the current query is the k-th gate of C. If $k = p_{\alpha_i}$, then we know that $\pi^n(\alpha_i) = \beta$, so we immediately terminate this iteration, and proceed to the next iteration. Otherwise we know that we should return 0 to this query.

Whenever C makes a query to $\mathrm{GapMINKT}^{\Pi}$, namely $\mathrm{GapMINKT}^{\Pi}[x, 1^s, 1^t]$, we reply 1 if there is a Turing machine M satisfying the following requirements.

- The description length of M is at most $s + (c/3)\log t$, M runs in at most $\sqrt{c}t$ time, and M outputs x.
- M has access to a Π_{-i} oracle, but for every query $\Pi_{-i}[\alpha, \beta]$ it asks, either $\alpha \preceq \alpha_i$ or $\alpha \in \mathcal{S}$.

It is easy to see that if we reply 1, then $\mathrm{K}^{\sqrt{c}t, \Pi_{-i}}(x) \le s+(c/3)\log t$, therefore by Lemma 6.2, $(x, 1^s, 1^t)$ is not a NO instance of $\mathrm{GapMINKT}^{\Pi}$. On the other hand, if x is a YES instance of $\mathrm{GapMINKT}^{\Pi}$, then again by Lemma 6.2, we have $\mathrm{K}^{ct\log t, \Pi_{-i}}(x) \le s+c\log t$. By our construction of $\mathcal{S}$, there is a Turing machine M satisfying the above conditions, hence we reply 1. Therefore, queries to $\mathrm{GapMINKT}^{\Pi}$ are also answered correctly.

As every oracle gate of C on input α_i are answered correctly, we can successfully recover $C(\alpha_i) = \pi^n(\alpha_i)$.

The compressed length of π^n. The number of bits we need to describe π^n, including C is

$$\log\left(\binom{2^n}{|\mathcal{S}|}^2 |\mathcal{S}|!\right) + (2^n - |\mathcal{S}|)\log|C| + |C|\log|C|. \tag{12}$$

Now we upper bound $|\mathcal{S}|$. There are $2^{0.9n}$ inputs on which C is correct; these are the inputs not in $\mathcal{S}$ initially. Each time we perform the i-th iteration for some i such that $\alpha_i \notin \mathcal{S}$, we add at most $|C|(\sqrt{c}+1)$ inputs to $\mathcal{S}$. This is because C can make at most $|C|$ queries to Π, and the sum of $\sqrt{c}t$ over all queries $\mathrm{GapMINKT}^{\Pi}[x, 1^s, 1^t]$ made by C is at most $\sqrt{c}|C|$. Put in other words, each time we ensure that a new element (α_i) is not in the final $\mathcal{S}$, we add at most $|C|(\sqrt{c}+1)$ elements into $\mathcal{S}$. Therefore

$$2^n - |\mathcal{S}| \ge \Omega\left(\frac{2^{0.9n}}{|C|}\right) \ge \Omega(2^{0.79n}).$$

Now let $2^n - |\mathcal{S}| = 2^{\lambda n}$, then $\lambda \geq 0.78$. The number of bits we save is

$$\begin{aligned}
\log((2^n)!) - (12) \geq& \log((2^{\lambda n})!) - \log \binom{2^n}{2^{\lambda n}} - (2^{\lambda n} + |C|) \log |C| \\
\geq& 2^{\lambda n}(\lambda n - \log e) - 2^{\lambda n}(2 - (\lambda - 1)n) - (2^{\lambda n} + |C|) \log |C| - \Theta(n) \qquad (13) \\
\geq& 2^{\lambda n}((2\lambda - 1)n - 4) - (2^{\lambda n} + |C|) \log |C| - \Theta(n) \\
\geq& 2^{0.78n}(0.56n - 4) - 2^{0.78n} \cdot 0.11n + \Theta(n) \qquad (14) \\
\geq& 0.4n \cdot 2^{0.78n}.
\end{aligned}$$

Here, Eq. (13) uses the fact that $\log \binom{n}{k} \leq k(2 - \log(k/n))$ (Claim 2.17), and $\log(n!) = n(\log n - \log e) + \Theta(\log n)$; and Eq. (14) uses the fact that $\lambda \geq 0.78$ and $|C| \leq 2^{0.1n}$.

Putting it together. Let ω be an infinite binary string that satisfies the promise of Theorem 2.13. That is, for almost every integer n, $\mathrm{C}(\omega_{1\sim n} \mid n) \geq n - 2\log n$. Let $\ell_n = \lfloor \log((2^n)!) \rfloor$, then the number of permutations $\pi^n : \{0,1\}^n \to \{0,1\}^n$ is in the range of $[2^{\ell_n}, 2^{\ell_n+1})$. Let $\ell_{\leq n} = \sum_{i=1}^{n} \ell_i$, then $\ell_{\leq n} \leq 3^n$ for large enough n. We define π^n as the permutation decoded from the string

$$\omega_{(\ell_{\leq n-1}+1)\sim \ell_{\leq n}}.$$

For any large enough n, suppose that there is a circuit of size $2^{0.005n}$ with Π and GapMINKT gates that computes L^{hard} on a $(1/2 + 1/2^{0.005n})$ fraction of inputs. Then there is a circuit C of size $2^{0.1n}$ with Π and GapMINKT gates that inverts π^n on a $2^{-0.1n}$ fraction of inputs. Let $N = \ell_{\leq 2^{0.1n}}$, we will compress $\omega_{1\sim N}$ into less than $N - 2\log N$ bits.

We first write down every permutation $\pi_{n'}$ in verbatim, where $n' \leq 2^{0.1n}$ and $n' \neq n$. Then we use the above method to compress π_n into $\log((2^n)!) - 0.4n \cdot 2^{0.78n}$ bits. Note that since C only have access to Π gates of fanin at most $2^{0.1n}$, given the information of every $\pi_{n'}$ ($n' \leq 2^{0.1n}$ and $n' \neq n$), we can recover π_n from these $\log((2^n)!) - 0.4n \cdot 2^{0.78n}$ bits. It follows that we can compress $\omega_{1\sim N}$ into less than $N - 0.4n \cdot 2^{0.78n} < N - 2\log N$ bits, contradicting the Kolmogorov randomness of ω.

In conclusion, in our oracle world, any circuit of size $2^{o(n)}$ cannot $(1/2+1/2^{-o(n)})$-approximate L^{hard} even with a GapMINKT oracle gate. □

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009. (cit. on p. 10)

[AB18] Barış Aydınlıoğlu and Eric Bach. Affine relativization: Unifying the algebrization and relativization barriers. *ACM Transactions on Computation Theory*, 10(1):1:1–1:67, 2018. doi:10.1145/3170704. (cit. on p. 4)

[ABK+06] Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM Journal of Computing*, 35(6):1467–1493, 2006. doi:10.1137/050628994. (cit. on p. 1, 2, 4, 5, 10, 24)

[AH19] Eric Allender and Shuichi Hirahara. New insights on the (non-)hardness of circuit minimization and related problems. *ACM Transactions on Computation Theory*, 11(4):27:1–27:27, 2019. doi:10.1145/3349616. (cit. on p. 2)

[AIV19] Eric Allender, Rahul Ilango, and Neekon Vafa. The non-hardness of approximating circuit size. In *Proc. 14th International Computer Science Symposium in Russia (CSR)* , volume 11532 of *Lecture Notes in Computer Science*, pages 13–24, 2019. doi:10.1007/978-3-030-19955-5_2. (cit. on p. 2)

[AKRR11] Eric Allender, Michal Koucký, Detlef Ronneburger, and Sambuddha Roy. The pervasive reach of resource-bounded Kolmogorov complexity in computational complexity theory. *Journal of Computer and System Sciences*, 77(1):14–40, 2011. doi:10.1016/j.jcss.2010.06.004. (cit. on p. 2)

[AW09] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Transactions on Computation Theory*, 1(1):2:1–2:54, 2009. doi:10.1145/1490270.1490272. (cit. on p. 4, 24)

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991. doi:10.1007/BF01200056. (cit. on p. 4, 24)

[BFNW93] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computatioanl Complexity*, 3:307–318, 1993. doi:10.1007/BF01275486. (cit. on p. 4, 24)

[BGS75] Theodore P. Baker, John Gill, and Robert Solovay. Relativizations of the P =?NP question. *SIAM Journal of Computing*, 4(4):431–442, 1975. doi:10.1137/0204037. (cit. on p. 1)

[CIKK16] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In *Proc. 31st Computational Complexity Conference (CCC)*, volume 50 of *LIPIcs*, pages 10:1–10:24, 2016. doi:10.4230/LIPIcs.CCC.2016.10. (cit. on p. 1, 2, 17)

[CKLM20] Mahdi Cheraghchi, Valentine Kabanets, Zhenjian Lu, and Dimitrios Myrisiotis. Circuit lower bounds for MCSP from local pseudorandom generators. *ACM Transactions on Computation Theory*, 12(3):21:1–21:27, 2020. doi:10.1145/3404860. (cit. on p. 8)

[Fu20] Bin Fu. Hardness of sparse sets and minimal circuit size problem. In *Proc. 26th International Computing and Combinatorics Conference (COCOON)* , volume 12273 of *Lecture Notes in Computer Science*, pages 484–495, 2020. doi:10.1007/978-3-030-58150-3_39. (cit. on p. 2)

[GGKT05] Rosario Gennaro, Yael Gertner, Jonathan Katz, and Luca Trevisan. Bounds on the efficiency of generic cryptographic constructions. *SIAM Journal of Computing*, 35(1):217–246, 2005. doi:10.1137/S0097539704443276. (cit. on p. 9, 30, 40)

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proc. 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 25–32, 1989. doi:10.1145/73007.73010. (cit. on p. 14)

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991. doi:10.1145/116825.116852. (cit. on p. 3, 5)

[Gol08] Oded Goldreich. *Computational complexity: a conceptual perspective*. Cambridge University Press, 2008. doi:10.1017/CBO9780511804106. (cit. on p. 10)

[Hir18] Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *Proc. 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 247–258, 2018. doi:10.1109/FOCS.2018.00032. (cit. on p. 1, 2, 3, 5, 9, 17, 32, 37)

[Hir20a] Shuichi Hirahara. Characterizing average-case complexity of PH by worst-case meta-complexity. In *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 50–60, 2020. (cit. on p. 1, 3, 32, 37)

[Hir20b] Shuichi Hirahara. Non-disjoint promise problems from meta-computational view of pseudorandom generator constructions. In *Proc. 35th Computational Complexity Conference (CCC)*, volume 169 of *LIPIcs*, pages 20:1–20:47, 2020. doi:10.4230/LIPIcs.CCC.2020.20. (cit. on p. 37)

[Hir20c] Shuichi Hirahara. Unexpected hardness results for Kolmogorov complexity under uniform reductions. In *Proc. 52nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 1038–1051, 2020. doi:10.1145/3357713.3384251. (cit. on p. 1, 3)

[Hir21] Shuichi Hirahara. Average-case hardness of NP from exponential worst-case hardness assumptions. In *Proc. 53rd Annual ACM Symposium on Theory of Computing (STOC)*, 2021. To appear. URL: https://eccc.weizmann.ac.il/report/2021/058. (cit. on p. 1)

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. URL: http://www.jstor.org/stable/2282952. (cit. on p. 14)

[HP15] John M. Hitchcock and Aduri Pavan. On the NP-completeness of the minimum circuit size problem. In *Proc. 35th Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 45 of *LIPIcs*, pages 236–245, 2015. doi:10.4230/LIPIcs.FSTTCS.2015.236. (cit. on p. 2)

[HS17] Shuichi Hirahara and Rahul Santhanam. On the average-case complexity of MCSP and its variants. In *Proc. 32nd Computational Complexity Conference (CCC)*, volume 79 of *LIPIcs*, pages 7:1–7:20, 2017. doi:10.4230/LIPIcs.CCC.2017.7. (cit. on p. 1, 3, 5, 8, 11, 12, 21)

[HW16] Shuichi Hirahara and Osamu Watanabe. Limits of minimum circuit size problem as oracle. In *Proc. 31st Computational Complexity Conference (CCC)*, volume 50 of *LIPIcs*, pages 18:1–18:20, 2016. doi:10.4230/LIPIcs.CCC.2016.18. (cit. on p. 2, 10)

[IKK09] Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. An axiomatic approach to algebrization. In *Proc. 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 695–704, 2009. doi:10.1145/1536414.1536509. (cit. on p. 4)

[Ila20a] Rahul Ilango. Approaching MCSP from above and below: Hardness for a conditional variant and $\mathsf{AC}^0[p]$. In *Proc. 11th Conference on Innovations in Theoretical Computer Science (ITCS)*, volume 151 of *LIPIcs*, pages 34:1–34:26, 2020. doi:10.4230/LIPIcs.ITCS.2020.34. (cit. on p. 2)

[Ila20b] Rahul Ilango. Connecting perebor conjectures: Towards a search to decision reduction for minimizing formulas. In *Proc. 35th Computational Complexity Conference (CCC)*, volume 169 of *LIPIcs*, pages 31:1–31:35, 2020. doi:10.4230/LIPIcs.CCC.2020.31. (cit. on p. 2)

[Ila20c] Rahul Ilango. Constant depth formula and partial function versions of MCSP are hard. In *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 424–433, 2020. (cit. on p. 2)

[ILO20] Rahul Ilango, Bruno Loff, and Igor Carboni Oliveira. NP-hardness of circuit minimization for multi-output functions. In *Proc. 35th Computational Complexity Conference (CCC)*, volume 169 of *LIPIcs*, pages 22:1–22:36, 2020. doi:10.4230/LIPIcs.CCC.2020.22. (cit. on p. 2)

[Imp95] Russell Impagliazzo. A personal view of average-case complexity. In *Proc. 10th Annual Structure in Complexity Theory Conference*, pages 134–147, 1995. doi:10.1109/SCT.1995.514853. (cit. on p. 26)

[IW01] Russell Impagliazzo and Avi Wigderson. Randomness vs time: Derandomization under a uniform assumption. *Journal of Computer and System Sciences*, 63(4):672–688, 2001. doi:10.1006/jcss.2001.1780. (cit. on p. 1)

[KC00] Valentine Kabanets and Jin-Yi Cai. Circuit minimization problem. In *Proc. 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 73–79, 2000. doi:10.1145/335305.335314. (cit. on p. 1, 2, 3, 10)

[Ko91] Ker-I Ko. On the complexity of learning minimum time-bounded Turing machines. *SIAM Journal of Computing*, 20(5):962–986, 1991. doi:10.1137/0220059. (cit. on p. 1, 9, 11)

[Lev] Leonid A. Levin. Hardness of search problems. Accessed 12-June-2021. URL: https://www.cs.bu.edu/fac/lnd/research/hard.htm. (cit. on p. 2)

[Lev73] Leonid A. Levin. Universal sequential search problems. *Problemy peredachi informatsii*, 9(3):115–116, 1973. (cit. on p. 2)

[Lev84] Leonid A. Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61(1):15–37, 1984. doi:10.1016/S0019-9958(84)80060-1. (cit. on p. 10)

[LP20] Yanyi Liu and Rafael Pass. On one-way functions and Kolmogorov complexity. In *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1243–1254, 2020. (cit. on p. 1, 3, 5)

[LV08] Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications, Third Edition*. Texts in Computer Science. Springer, 2008. doi:10.1007/978-0-387-49820-1. (cit. on p. 13)

[MW17] Cody D. Murray and R. Ryan Williams. On the (non) NP-hardness of computing circuit complexity. *Theory of Computing*, 13(1):1–22, 2017. doi:10.4086/toc.2017.v013a004. (cit. on p. 2)

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994. doi:10.1016/S0022-0000(05)80043-1. (cit. on p. 1)

[Oli19] Igor Carboni Oliveira. Randomness and intractability in Kolmogorov complexity. In *Proc. 46th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 132 of *LIPIcs*, pages 32:1–32:14, 2019. doi:10.4230/LIPIcs.ICALP.2019.32. (cit. on p. 2)

[OS17] Igor Carboni Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *Proc. 32nd Computational Complexity Conference (CCC)*, volume 79 of *LIPIcs*, pages 18:1–18:49, 2017. doi:10.4230/LIPIcs.CCC.2017.18. (cit. on p. 1)

[OW93] Rafail Ostrovsky and Avi Wigderson. One-way functions are essential for non-trivial zero-knowledge. In *Proc. Second Israel Symposium on Theory of Computing Systems, (ISTCS)*, pages 3–17, 1993. doi:10.1109/ISTCS.1993.253489. (cit. on p. 5, 12)

[RR97] Alexander A. Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997. doi:10.1006/jcss.1997.1494. (cit. on p. 1, 3, 11)

[RS21] Hanlin Ren and Rahul Santhanam. Hardness of KT characterizes parallel cryptography. In *Proc. 36th Computational Complexity Conference (CCC)*, 2021. To appear. URL: https://eccc.weizmann.ac.il/report/2021/057. (cit. on p. 1)

[San20] Rahul Santhanam. Pseudorandomness and the minimum circuit size problem. In *Proc. 11th Conference on Innovations in Theoretical Computer Science (ITCS)*, volume 151 of *LIPIcs*, pages 68:1–68:26, 2020. doi:10.4230/LIPIcs.ITCS.2020.68. (cit. on p. 1, 3, 5, 8, 17)

[SS20] Michael Saks and Rahul Santhanam. Circuit lower bounds from NP-hardness of MCSP under Turing reductions. In *Proc. 35th Computational Complexity Conference (CCC)*, volume 169 of *LIPIcs*, pages 26:1–26:13, 2020. doi:10.4230/LIPIcs.CCC.2020.26. (cit. on p. 2)

[Tra84] Boris A. Trakhtenbrot. A survey of Russian approaches to perebor (brute-force searches) algorithms. *IEEE Annals of the History of Computing*, 6(4):384–400, 1984. doi:10.1109/MAHC.1984.10036. (cit. on p. 1)

[TV07] Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007. doi:10.1007/s00037-007-0233-x. (cit. on p. 1)

[Vad06] Salil P. Vadhan. An unconditional study of computational zero knowledge. *SIAM Journal of Computing*, 36(4):1160–1214, 2006. doi:10.1137/S0097539705447207. (cit. on p. 12)

[Vad12] Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012. doi:10.1561/0400000010. (cit. on p. 28)

[Wee06] Hoeteck Wee. Finding Pessiland. In *Proc. 3rd Theory of Cryptography Conference (TCC)* , volume 3876 of *Lecture Notes in Computer Science*, pages 429–442, 2006. doi:10.1007/11681878_22. (cit. on p. 8, 9, 26, 29, 31)

[Wil85] Christopher B. Wilson. Relativized circuit complexity. *Journal of Computer and System Sciences*, 31(2):169–181, 1985. doi:10.1016/0022-0000(85)90040-6. (cit. on p. 10)

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il