Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 9 (2021)

One-way Functions and a Conditional Variant of MKTP

Eric Allender[*] Mahdi Cheraghchi[†] Dimitrios Myrisiotis[‡] Harsha Tirumala[§]
Ilya Volkovich[¶]

April 18, 2021

Abstract

One-way functions (OWFs) are central objects of study in cryptography and computational complexity theory. In a seminal work, Liu and Pass (FOCS 2020) proved that the average-case hardness of computing time-bounded Kolmogorov complexity is *equivalent* to the existence of OWFs. It remained an open problem to establish such an equivalence for the average-case hardness of some NP-complete problem. In this paper, we make progress on this question by studying a conditional variant of the Minimum KT-complexity Problem (MKTP), which we call McKTP, as follows.

1. First, we prove that if McKTP is average-case hard on a polynomial fraction of its instances, then there exist OWFs.
2. Then, we observe that McKTP is NP-complete under polynomial-time randomized reductions. That is, there *are* NP-complete problems whose average-case hardness implies the existence of OWFs.
3. Finally, we prove that the existence of OWFs implies the nontrivial average-case hardness of McKTP.

Thus the existence of OWFs is inextricably linked to the average-case hardness of this NP-complete problem.

Keywords. Kolmogorov complexity, Minimum KT-complexity Problem, MKTP, Minimum Conditional KT-complexity Problem, McKTP, one-way functions, OWFs, average-case hardness, pseudorandom generators, PRGs, pseudorandom functions, PRFs, distinguishers, learning algorithms, reductions

1 Introduction

One-way functions (OWFs) —that is, functions that are easy to compute but hard to invert— are objects of great importance in cryptography and computational complexity. For example, it is known that OWFs exist if and only if pseudorandom generators exist [HILL99] and, moreover, if OWFs exist, then $\mathsf{P} \neq \mathsf{NP}$.

In this paper, we ask the following question: *Can the existence of OWFs be based on the average-case hardness of some* NP*-complete problem?* We take concrete steps toward giving an affirmative answer to this question, by presenting a candidate problem.

Of course, if there is any problem in NP that is hard on average, it follows that there is an NP-complete problem that also shares this property; see Proposition A.1. Thus, our main contribution is to present an NP-complete problem whose average-case complexity is tightly linked to the existence of OWFs.

[*] Department of Computer Science, Rutgers University, Piscataway, NJ, USA; E-mail: allender@cs.rutgers.edu.
[†] Department of EECS, University of Michigan, Ann Arbor, MI, USA; E-mail: mahdich@umich.edu.
[‡] Department of Computing, Imperial College London, London, UK; E-mail: d.myrisiotis17@ic.ac.uk.
[§] Department of Computer Science, Rutgers University, Piscataway, NJ, USA; E-mail: hs675@scarletmail.rutgers.edu.
[¶] Computer Science Department, Boston College, MA, USA; E-mail: ilya.volkovich@bc.edu.

ISSN 1433-8092

1.1 Prior work

An early goal in cryptographic research was to base the existence of cryptographically-secure one-way functions on the worst-case complexity of some NP-complete problem. This goal remains elusive; it was shown in [AGGM06] that no black-box argument of this sort can proceed based on non-adaptive reductions. Non-adaptive worst-case-to-average-case reductions were also studied by Bogdanov and Trevisan [BT06b], who showed that such reductions to sets in NP exist only for problems in $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$. Recent work by Nanashima [Nan21] holds open the possibility that the security of OWFs can be based on an *adaptive* black-box reduction, by first establishing a non-adaptive black-box reduction basing the existence of *auxiliary input one-way functions* on the worst-case complexity of an NP-complete problem, although this would also require non-relativizing techniques. Instead of worst-case hardness, the focus of our work is on average-case hardness assumptions. A nice survey on this area, that lays out many of the issues about one-way functions and average-case complexity, is the one by Bogdanov and Trevisan [BT06a].

Recently, Liu and Pass showed that OWFs exist if and only if computing the time-bounded Kolmogorov complexity of strings is average-case hard [LP20]. This is an important result, but it falls short of basing the existence of OWFs on an NP-complete problem, since

1. computing the time-bounded Kolmogorov complexity is not known to be NP-hard, and

2. the computation of this function can be done in polynomial time with an NP-oracle, but it does not translate directly to the average-case complexity of a language *in* NP.

The notion of time-bounded Kolmogorov complexity considered in [LP20] requires that the time used by the universal Turing machine producing x from description Π must be at least $|x|$, in contrast to the notion of time-bounded Kolmogorov complexity (KT) that we study here, where the running time can be as small as $\log |x|$. In more recent work [LP21a], Liu and Pass consider a notion of time-bounded Kolmogorov complexity that is more closely related to KT, where the running time of the universal machine must be at least the length of the description Π, but can be much less than $|x|$.

Santhanam [San20] showed that a restricted type of hitting-set generator exists if and only if the Minimum Circuit Size Problem (MCSP) is zero-error average-case hard. Hirahara also proved similar results connecting the worst-case and the zero-error average-case complexity of problems related to MCSP and Kolmogorov complexity [Hir18].

More recently, Brzuska and Couteau [BC20] discuss basing OWFs on average-case hardness, noting that it remains an open question to do this for the general notion of average-case hardness; they do not consider zero-error average-case hardness as studied by Hirahara and Santhanam [HS17]. They present some negative results, indicating the difficulty of establishing the existence of fine-grained one-way functions, based on the existence of average-case hardness, via black-box reductions.

1.2 Our results

We connect the existence of OWFs to the average-case hardness of computing an NP-complete variant of MKTP. Initially, we prove that the average-case hardness of a conditional variant of MKTP, which we term McKTP, implies the existence of OWFs.

Theorem 1.1 (Informal)**.** *OWFs exist if* McKTP *is hard-on-average on a polynomial fraction of its instances.*

Theorem 1.1 should be contrasted to the recent work by Liu and Pass [LP20], where they prove that the average-case hardness of computing the time-bounded Kolmogorov complexity K^t of a string is equivalent to the existence of OWFs. Moreover, Theorem 1.1 is complemented by proving that McKTP is NP-complete under randomized reductions.

Theorem 1.2 (Informal)**.** McKTP *is* NP*-complete under polynomial-time randomized one-side-error reductions.*

Theorem 1.1 and Theorem 1.2 together answer in the affirmative the open question of whether there exists some NP-complete problem whose average-case hardness implies the existence of OWFs.

Moreover, Theorem 1.1 suggests an approach for excluding Impagliazzo's *Pessiland* [Imp95], that is, a version of our world where there are average-case hard problems in NP *and* there are no OWFs. This approach is based on the following observation. If McKTP is NP-hard under average-case reductions, then by Theorem 1.1 the existence of an average-case hard problem in NP would imply the existence of OWFs. Therefore proving that McKTP is NP-hard under average-case reductions excludes Pessiland.

Finally, we prove a *weak* converse of Theorem 1.1.

Theorem 1.3 (Informal)**.** *OWFs exist only if* McKTP *is hard-on-average on an* exponential *fraction of its instances.*

1.2.1 How significant are our results?

The reader may wonder whether the hypothesis of Theorem 1.1 is overly strong. Is there perhaps some trivial heuristic that succeeds well on average for this NP-complete decision problem?

The input to the problem consists of a triple (x, y, θ), where the question is whether $\mathrm{KT}(x \mid y) \leq \theta$, where θ is a number bounded by $|x| + O(\log |x|)$. A simple heuristic is to accept if θ is at the high end of this range, and reject otherwise; one can augment this to accept for slightly lower values of θ if x has certain hallmarks of low complexity (such as starting or ending with a logarithmic number of zeros, or agreeing with y on those substrings. However, when inputs are chosen at random, this heuristic still seems likely to fail with constant probability if θ is close to the boundary between where the heuristic accepts and rejects. In particular, it is far from clear how to design a heuristic that would have failure probability less than, say $1/s^2$, where θ ranges over a domain of size s. In particular, it seems quite plausible that there is a constant k for which no heuristic can achieve failure probability less than $1/s^k$, which is precisely the hypothesis of Theorem 1.1, and is sufficient for the existence of OWFs.

By the same token, the conclusion of Theorem 1.3 gives a much weaker, but still non-trivial, average-case hardness condition for McKTP. Taken together, these results for the first time draw explicit connections between the average-case hardness of a specific decision problem in NP and the existence of cryptographically-secure one-way functions.

1.3 Our techniques

Our main results are Theorem 1.1, Theorem 1.2, and Theorem 1.3. Below we provide some intuition regarding their proofs.

1. Theorem 1.1 is proved by

   (a) giving an average-case decision-to-search reduction for McKTP (see Lemma 3.3) and

   (b) observing that a recent result by Liu and Pass [LP20], whereby they prove that the average-case hardness of a search variant of time-bounded Kolmogorov complexity K^t yields OWFs, can be adjusted to the case of McKTP as well (see Lemma 3.4).

   The three properties of time-bounded Kolmogorov complexity K^t, for some $t : \mathbb{N} \to \mathbb{N}$ where $t(n) \geq n$ for all $n \in \mathbb{N}$, that are used by Liu and Pass, are as follows.

   i. One can create a string of low time-bounded Kolmogorov complexity in polynomial time. This can be done by running a universal Turing machine U on some string, for polynomially-many steps, and subsequently recording the output of U.

   ii. For any string x, the possible values of its K^t complexity are polynomially-many in $|x|$. In fact, there is a $c > 0$ such that, for any function $t : \mathbb{N} \to \mathbb{N}$ such that $t(n) \geq n$ for all $n \in \mathbb{N}$, and any string x, the possible values of $K^t(x)$ are at most $|x| + c$.

   iii. The following domination property holds. Let $x^* \in \{0,1\}^n$ be a string, and $c > 0$ be as in Item 1(b)ii. Then,

$$\Pr_{\Pi \sim \{0,1\}^{n+c}} \left[U\left(\Pi, 1^{t(n)}\right) = x^* \right] \geq \frac{1}{2^{n+c}} = \frac{2^{-n}}{2^c} \geq \frac{\mathbf{Pr}_{x \sim \{0,1\}^n}[x = x^*]}{\mathrm{poly}(n)}.$$

   As it turns out, all of these properties are satisfied even when one considers McKTP. This is true for the following reasons.

   iv. One can create a string of low conditional KT complexity in polynomial time. As above, this can be done by running a universal *oracle* Turing machine U, that has oracle access to some string y, on a string x for $t \leq \mathrm{poly}(|x|)$ steps, and subsequently recording the output of U^y.

v. For any string x, the possible values of its KT complexity given any string y are polynomially-many in $|x|$. In fact, there is a $c > 0$ such that for any string x, the possible values of the conditional KT complexity of (x, y) are at most $|x| + c \log |x|$.

vi. The following domination property holds (for KT *and* conditional KT complexity). Let $x^* \in \{0,1\}^n$ be a string, $c > 0$ be as in Item 1(b)v, and $n_c := n + c \log n$. Then, for all $y \in \{0,1\}^*$, it is the case that

$$\begin{aligned}\Pr_{\substack{\Pi \sim \{0,1\}^{n_c},\ t \in [n_c]: \\ |\Pi| + t \leq n_c}} \left[\text{for all } 1 \leq i \leq n,\ U^{\Pi, y}(i, 1^t) = x_i^*\right] &\geq \frac{1}{2^{n_c + \log n_c}} \\ &= \frac{2^{-n}}{n^c\,(n + c \log n)} \\ &= \frac{\mathbf{Pr}_{x \sim \{0,1\}^n}[x = x^*]}{\mathrm{poly}(n)}.\end{aligned}$$

2. Theorem 1.2 is proved by

   (a) noting that McKTP is in NP (see Lemma 2.14) and

   (b) showing the NP-hardness of McKTP (see Corollary B.9). This is done by giving a polynomial-time randomized reduction from Set Cover, which is NP-hard to approximate (see Corollary B.8), to an appropriate gap version of McKTP; see Corollary B.7. Note that this step closely mimics the proof of Ilango [Ila20] for the NP-hardness of Minimum Oracle Circuit Size Problem (MOCSP).

3. Theorem 1.3 is proved by giving a proof of its contrapositive statement, as explained by the items below.

   (a) Assume that McKTP is easy on average under the uniform distribution.

   (b) By a corollary of Ilango, Loff, and Oliveira (see Lemma 2.23), for all $a \geq 1$, there exists a learning algorithm for $\mathsf{SIZE}[n^a]$ that works for infinitely many $n \in \mathbb{N}$; see Lemma 4.2.

   (c) By a learner-to-distinguisher reduction (see Lemma 4.5), for every polynomial-time computable Boolean function family $\{f_y\}_{y \in \{0,1\}^*}$, there is a distinguisher for $\{f_y\}_{y \in \{0,1\}^*}$.

   (d) By the correctness of the works by Håstad, Implagliazzo, Levin, and Luby [HILL99], and Goldreich, Goldwasser, and Micali [GGM86], there are no OWFs.

1.4 Discussion

Just as we were preparing to post this article, we were made aware that Liu and Pass [LP21b], working independently (and having seen an earlier version of our work, that had errors), now claim that a (slightly different) version of time-bounded conditional Kolmogorov complexity is

1. NP-complete under polynomial-time randomized reductions, and

2. hard on average on a *polynomial* fraction of its instances if and only if OWFs exist.

The main points of departure between the work by Liu and Pass and ours, are that Liu and Pass consider conditional K^t complexity, while we consider conditional KT complexity, and that their work claims an *equivalence* between the average-case hardness of an NP-complete problem and the existence of OWFs.

1.5 Paper organization

In Section 2 we give some background knowledge, and useful facts. We prove Theorem 1.1 in Section 3 and Theorem 1.3 in Section 4. Finally, we prove Theorem 1.2 in Appendix B.

2 Preliminaries

2.1 Sets and strings

We denote the natural numbers by $\mathbb{N}$ and the positive reals by $\mathbb{R}_{>0}$. For any $n \in \mathbb{N}$, we denote the set $\{1, \ldots, n\}$ by $[n]$. Let $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ be a string of length n; we write $|x| := n$. The empty string is denoted by λ.

2.2 Boolean functions

We denote by $\mathcal{F}_n$ the class of all Boolean functions on n variables. We identify infinite Boolean functions $f : \{0,1\}^* \to \{0,1\}$ with collections $\{f_n\}_{n\in\mathbb{N}}$, whereby $f_n : \{0,1\}^n \to \{0,1\}$ for all $n \in \mathbb{N}$.

2.3 Probability theory

We will use the following useful facts from probability theory.

Lemma 2.1 (Averaging argument)**.** *If $X \in [0,1]$ is a random variable with $\mu := \mathbf{E}[X]$, then for all $0 < c < 1$ it is the case that*

$$\mathbf{Pr}[X \geq c\mu] \geq (1-c)\,\mu.$$

Lemma 2.2 (Markov's inequality)**.** *If X is a non-negative random variable with $\mu := \mathbf{E}[X]$, then for all $k > 0$ it is the case that*

$$\mathbf{Pr}[X \geq k\mu] \leq \frac{1}{k}.$$

Lemma 2.3 (Chernoff bound)**.** *Let $n \in \mathbb{N}$ and $X_1, \ldots, X_n$ be Boolean random variables that are independent and identically distributed. Let $X := \sum_{i=1}^{n} X_i$ and $\mu := \mathbf{E}[X]$. Then, for all $0 \leq \delta \leq 1$, it is the case that*

$$\mathbf{Pr}[X \geq (1+\delta)\,\mu] \leq e^{-\frac{\delta^2\mu}{3}}.$$

2.4 Circuit complexity

We consider Boolean circuits over the bounded fan-in $\{\wedge_2, \vee_2, \neg, 0, 1\}$ basis. Given a circuit, its *size* is the number of its gates. Let $s : \mathbb{N} \to \mathbb{N}$ be a function. If we use s to upper bound the size of some circuit, then we shall call s a *size function.*

Lemma 2.4. *Let C be a circuit of size $s \in \mathbb{N}$. Then, C can be described by using $O(s \log s)$ bits.*

Lemma 2.5. *Let $s : \mathbb{N} \to \mathbb{N}$ be a size function. Then, there exists an algorithm that, on input a description $d_C \in \{0,1\}^*$ of a circuit C that has $n \in \mathbb{N}$ inputs and size $s(n)$, whereby $|d_C| \leq O(s(n) \log s(n))$ (as in Lemma 2.4), and a string $x \in \{0,1\}^n$, outputs $C(x) \in \{0,1\}$ in time $O\Big(s(n)^2 \log s(n)\Big)$.*

Proof. Let C be a circuit that has n inputs and size $s(n)$, and let $x \in \{0,1\}^n$ be a string. Then, on input x, any gate of C can be evaluated in a bottom-up fashion by parsing the description $d_C \in \{0,1\}^*$ of C a constant number of times. The idea described above can be implemented as an algorithm that runs in time

$$s(n) \cdot O(|d_C|) \leq s(n) \cdot O(s(n) \log s(n)) = O\Big(s(n)^2 \log s(n)\Big). \qquad \square$$

Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, the *circuit complexity of* f, denoted $\mathrm{CC}(f)$, is the size of a minimum size circuit that computes f. For a size function $s : \mathbb{N} \to \mathbb{N}$, we denote by $\mathsf{SIZE}[s(n)]$ the class of Boolean functions $f = \{f_n\}_{n\in\mathbb{N}}$, whereby $f_n : \{0,1\}^n \to \{0,1\}$ for all $n \in \mathbb{N}$, such that $\mathrm{CC}(f_n) \leq s(n)$ for all $n \in \mathbb{N}$.

We require the following trivial lower bound on the circuit complexity of an arbitrary Boolean function (which one can also regard as a convention regarding the measure CC).

Lemma 2.6. *Let $f : \{0,1\}^n \to \{0,1\}$ be a function. Then, $\mathrm{CC}(f) \geq n$.*

2.5 Uniform algorithms

In this work, we do not distinguish between Turing machines and algorithms. We say that an algorithm A is a *PPT algorithm* if A is a probabilistic polynomial-time algorithm. If A is a PPT algorithm that runs in time $p(n)$ for a polynomial p, then we denote by $A(x;r)$ the output of A on input $x \in \{0,1\}^*$ using random bits $r \in \{0,1\}^{p(|x|)}$. We say that an algorithm A is a *PPT oracle algorithm* if A is a PPT algorithm that has access to some oracle. If A is a PPT oracle algorithm that runs in time $p(n)$ for a polynomial p and has access to an oracle for a language $L \subseteq \{0,1\}^*$, then we denote by $A^L(x;r)$ the output of A^L on input $x \in \{0,1\}^*$ using random bits $r \in \{0,1\}^{p(|x|)}$.

We require the following observation, which upper bounds the circuit complexity of uniform computations.

Lemma 2.7 ([PF79])**.** *Let $T : \mathbb{N} \to \mathbb{N}$ be a function. Let $f : \{0,1\}^n \to \{0,1\}$ be a function computable by a Turing machine in time $O(T(n))$. Then, there exists a circuit of size $O(T(n)\log T(n))$ that computes f.*

2.6 KT complexity

2.6.1 A universal Turing machine

In what follows, we fix some *efficient* universal Turing machine (UTM) U. Let $y, \Pi, z \in \{0,1\}^*$ and $t \in \mathbb{N}$. The notation $U^{\Pi,y}(z, 1^t)$ denotes the output of U when U runs the program Π on input z for at most t steps, given that U has extended oracle access to program Π and standard oracle access to auxiliary string y.

Standard oracle access to auxiliary string y means that U has a standard oracle tape T_y of $\log|y|$ cells, and that in order to read a bit y_i of y, whereby $1 \le i \le |y|$, the machine U has to write $i \in \{0,1\}^{\log|y|}$ on T_y and then enter a question state. In the next step, the contents of T_y are erased and replaced by a bit b such that $b = y_i$.

One important aspect of our choice of U is that, for every auxiliary string $y \in \{0,1\}^*$ and $1 \le i \le \log|y|$, the oracle query $y_i \overset{?}{=} 1$ is such that *requires* time $\log|y|$, and can be implemented in time $O(\log|y|)$.

Extended oracle access to program Π means that U has a tape T_Π of $|\Pi|$ cells that contains Π, and the head of T_Π has *both* the ability to jump to an indexed location $1 \le i \le |\Pi|$ of T_Π, namely $T_\Pi[i] = \Pi_i$, *and* to move left and right on T_Π. Note that in the former case the index i is written in a separate tape of $\log|\Pi|$ cells, specifically allocated for that purpose. (So extended oracle access implies the existence of two tapes that help facilitate the oracle query.)

The notation $U^{\Pi,y}(z)$ denotes the output of U when U runs the program Π on input z, until Π halts (if this is the case, otherwise Π runs forever), whereby U has extended oracle access to Π and standard oracle access to y.

In this work, we will assume that whenever U is given oracle access to a program Π, this access will be *extended*, and whenever U is given oracle access to an auxiliary string y, this access will be *standard*. This is mainly to avoid unnecessary complications in the proof of Theorem 1.2 (where it is convenient to have sequential access to Π, while requiring that each query to y uses logarithmic time) while maintaining the trivial upper bound on KT complexity (see Lemma 2.8) which requires oracle access to Π.

We will also assume that the output of U will either be 1 or 0, on any input.

2.6.2 Definition of KT complexity, and some properties

Given strings $x, y \in \{0,1\}^*$, we define the *KT complexity of x given y*, denoted $\mathrm{KT}(x \mid y)$, to be the minimum value of $|\Pi| + t$ over programs $\Pi \in \{0,1\}^*$ and run-time bounds $t \in \mathbb{N}$ whereby for all $1 \le i \le |x|$ it is the case that $U^{\Pi,y}(i, 1^t) = x_i$.[1] For all strings $x \in \{0,1\}^*$, we define $\mathrm{KT}(x)$ to be equal to $\mathrm{KT}(x \mid \lambda)$.

Lemma 2.8 ([ABK+06])**.** *There is a $c > 0$ such that for all $x \in \{0,1\}^*$ it is the case that $\mathrm{KT}(x)$ is at most $|x| + c\log|x|$.*

Corollary 2.9. *There is a $c > 0$ such that for all $x, y \in \{0,1\}^*$ it is the case that $\mathrm{KT}(x \mid y)$ is at most $|x| + c\log|x|$.*

[1] Originally [ABK+06], $\mathrm{KT}(x \mid y)$ was defined with the additional requirement that, for $i = |x| + 1$, $U^{\Pi,y}(i, 1^t) = *$. We do not need that additional complication here, although our theorems would also hold using that definition.

Lemma 2.10. *Let $n, m \in \mathbb{N}$ be such that $n \le m$, let $\sigma > 0$, and let $s := n^{1/\sigma}$. Then, for all $y \in \{0,1\}^m$ there exists $x \in \{0,1\}^n$ such that $\mathrm{KT}(x \mid y) \le s$.*

Proof. This can be seen by setting $x := y_1 \cdots y_n \in \{0,1\}^n$. In this case, $\mathrm{KT}(x \mid y) \le c \log n \le n^{1/\sigma} = s$, where c is from Corollary 2.9. □

Lemma 2.11. *Let $n \in \mathbb{N}$ be sufficiently large, $m \in \mathbb{N}$, and $c > 1$. Then, it is the case that*

$$\Pr_{\substack{x \sim \{0,1\}^n, \\ y \sim \{0,1\}^m}} \left[\mathrm{KT}(x \mid y) \le n^{1/c} \right] = o(1)\,,$$

where x and y are independent and uniformly random.

Proof. We have that

$$\begin{aligned}
\Pr_{x,y}\left[\mathrm{KT}(x \mid y) \le n^{1/c}\right] &= \sum_y \Pr_x\left[\mathrm{KT}(x \mid y) \le n^{1/c}\right] \cdot \frac{1}{2^m} \\
&= \sum_y \frac{\left|\left\{x \in \{0,1\}^n \mid \mathrm{KT}(x \mid y) \le n^{1/c}\right\}\right|}{2^n} \cdot \frac{1}{2^m} \\
&\le \sum_y \frac{\left|\left\{\Pi \in \{0,1\}^* \mid |\Pi| \le n^{1/c}\right\}\right|}{2^n} \cdot \frac{1}{2^m},
\end{aligned}$$

since $\mathrm{KT}(x \mid y) \le n^{1/c}$ implies that there exists a program $\Pi \in \{0,1\}^*$ and a run-time bound $t \in \mathbb{N}$ such that $U^{\Pi,y}(i, 1^t) = x_i$ for all $1 \le i \le n$, and $|\Pi| < |\Pi| + t \le n^{1/c}$, or

$$\begin{aligned}
&= \frac{\left|\left\{\Pi \in \{0,1\}^* \mid |\Pi| \le n^{1/c}\right\}\right|}{2^n} \\
&\le \frac{2^{n^{1/c}+1} - 1}{2^n} \\
&\le \frac{2^{n^{1/c}+1}}{2^n} \\
&= o(1)\,.
\end{aligned}$$

□

2.6.3 KT complexity and circuit complexity

Lemma 2.12. *For all $f : \{0,1\}^n \to \{0,1\}$, $1 \le t \le 2^n$, and $x_1, \ldots, x_t \in \{0,1\}^n$, it is the case that*

$$\mathrm{KT}((f(x_1), \ldots, f(x_t)) \mid (x_1, \ldots, x_t)) \le O\left(\mathrm{CC}(f)^3\right).$$

Proof. Let C_f be a minimum-size circuit that computes f. Let Π be a program as follows:

> On input $1 \le i \le n$, compute and output $C_f(x_i)$.

By Lemma 2.4, we can assume that the description of C_f has size $O(\mathrm{CC}(f) \log \mathrm{CC}(f))$, so we have that $|\Pi| \le O(\mathrm{CC}(f) \log \mathrm{CC}(f))$. By Lemma 2.5 and Lemma 2.6, the running time of Π is at most

$$O(\log n) + O(n \log n) + O\left(\mathrm{CC}(f)^2 \log \mathrm{CC}(f)\right) \le O\left(\mathrm{CC}(f)^3\right).$$

Therefore, $\mathrm{KT}((f(x_1), \ldots, f(x_t)) \mid (x_1, \ldots, x_t)) \le O\left(\mathrm{CC}(f)^3\right)$. □

2.7 Minimum Conditional KT-complexity Problem, and variants

We give here formal definitions of the computational problems that we will consider in this work. These are the decision and search variants of McKTP.

2.7.1 Decision variant

Definition 2.13. *Let $c > 0$ be as in Corollary 2.9. Let $n \in \mathbb{N}$ and $m : \mathbb{N} \to \mathbb{N}$. The* Minimum m-Conditional KT-complexity Problem of dimension n (McKT^{m}P of dimension n) *is defined as follows.*

- *Input: Strings $x \in \{0,1\}^n$, $y \in \{0,1\}^{m(n)}$, and a parameter $0 \leq \theta \leq n + c \log n$ in binary.*
- *Question: Is there a program $\Pi \in \{0,1\}^*$ and a run-time bound $t \in \mathbb{N}$ such that*

  $$U^{\Pi,y}(i, 1^t) = x_i$$

  for all $1 \leq i \leq n$, and $|\Pi| + t \leq \theta$?

The following result is a standard observation.

Lemma 2.14. *For all polynomial-time computable functions $m : \mathbb{N} \to \mathbb{N}$, it is the case that* McKT^{m}P *of dimension n is in* NP.

2.7.2 Search variant

Definition 2.15. *Let $n \in \mathbb{N}$ and $m : \mathbb{N} \to \mathbb{N}$. The* search variant of Minimum m-Conditional KT-complexity Problem of dimension n (Search McKT^{m}P of dimension n) *is defined as follows.*

- *Input: Strings $x \in \{0,1\}^n$ and $y \in \{0,1\}^{m(n)}$.*
- *Output: A program $\Pi \in \{0,1\}^*$ and a run-time bound $t \in \mathbb{N}$ in binary such that*

  $$U^{\Pi,y}(i, 1^t) = x_i$$

  for all $1 \leq i \leq n$, and the sum $|\Pi| + t$ is minimized over the choices of Π and t.

2.8 One-way functions

In the following, a function μ is said to be *negligible* if for every polynomial p there exists a $n_0 \in \mathbb{N}$ such that for all naturals $n > n_0$ it is the case that $\mu(n) \leq 1/p(n)$.

Definition 2.16. *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable function. We say that f is a* one-way function (OWF) *if for every PPT algorithm A there exists a negligible function μ such that for all $n \in \mathbb{N}$ it is the case that*

$$\Pr_{x \sim \{0,1\}^n, r} \left[A(1^n, f(x)\,; r) \in f^{-1}(f(x)) \right] < \mu(n)$$

where the size of r is equal to the running time of A.

We will also employ the following weaker notion of OWFs.

Definition 2.17. *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable function. We say that f is an* α-weak one-way function (α-weak OWF) *if for every PPT algorithm A and all sufficiently large $n \in \mathbb{N}$ it is the case that*

$$\Pr_{x \sim \{0,1\}^n, r} \left[A(1^n, f(x)\,; r) \in f^{-1}(f(x)) \right] < 1 - \alpha(n)$$

where the size of r is equal to the running time of A. We say that f is a weak one-way function (weak OWF) *if there exists some polynomial $q > 0$ such that f is a $(1/q)$-weak OWF.*

Yao [Yao82] proved that the existence of weak OWFs implies the existence of OWFs.

Theorem 2.18 ([Yao82])**.** *Assume that there exists a weak one-way function. Then there exists a one-way function.*

2.9 Average-case hardness/easiness

2.9.1 Decision problems

A *heuristic* H is a PPT algorithm that, on input any $x \in \{0,1\}^n$, outputs a value in $\{0,1\}$ along each computation path.

Definition 2.19 (Average-case hardness)**.** *Let $\alpha : \mathbb{N} \to [0,1]$ be a failure parameter function. We say that a function $f : \{0,1\}^n \to \{0,1\}$ is* α-hard-on-average (α-HoA) *if for all heuristics H and all sufficiently large $n \in \mathbb{N}$ it is the case that*

$$\Pr_{x \sim \{0,1\}^n, r}[H(x;r) = f(x)] \leq 1 - \alpha(n)$$

where the size of r is equal to the running time of H.

Definition 2.20 (Average-case easiness)**.** *Let $\alpha : \mathbb{N} \to [0,1]$ be a success parameter function. We say that a function $f : \{0,1\}^n \to \{0,1\}$ is* α-easy-on-average (α-EoA) *if f is not $(1-\alpha)$-hard-on-average.*

2.9.2 Search problems

Let $R \subseteq \{0,1\}^n \times \{0,1\}^*$ be a search problem. A *heuristic* H is a PPT algorithm that, on input any $x \in \{0,1\}^n$, outputs a value in $\{0,1\}^*$ along each computation path.

The notions of average-case hardness and easiness for search problems are defined in a fashion similar to that of decision problems; see Definition 2.19 and Definition 2.20.

2.9.3 Infinitely-often average-case easiness of McKTP, with advantage

We require the following definition, which is inspired by Ilango, Loff, and Oliveira [ILO20].

Definition 2.21 (Following Ilango, Loff, and Oliveira [ILO20])**.** *Let $m : \mathbb{N} \to \mathbb{N}$, $s : \mathbb{N} \to \mathbb{N}$ be such that $s(n) < n/4$ for all $n \in \mathbb{N}$, and $\alpha : \mathbb{N} \to [0,1]$. A probabilistic algorithm B* solves McKT^{m}P of dimension n with advantage α for a size parameter s and infinitely many $n \in \mathbb{N}$ *if, for infinitely many $n \in \mathbb{N}$, it is the case that*

$$\left| \Pr_{y,r}[B(1^n, x, y; r) = 1] - \Pr_{z,y,r}[B(1^n, z, y; r) = 1] \right| \geq \alpha(n)$$

where $y \in \{0,1\}^{m(n)}$ and $z \in \{0,1\}^n$ are uniformly random, $x = x(y) \in \{0,1\}^n$ is such that $\mathrm{KT}(x \mid y) \leq s$ (which exists by Lemma 2.10), and $r \in \{0,1\}^*$ has size equal to the running time of B. In this case, we may also say that McKT^{m}P *of dimension n is* easy-on-average with advantage α for a size parameter s and infinitely many $n \in \mathbb{N}$ (EoA with advantage α for a size parameter s and infinitely many $n \in \mathbb{N}$).

2.10 Learning algorithms

For a Boolean function $f : \{0,1\}^n \to \{0,1\}$, an *example oracle for f*, denoted $\mathrm{EX}(f)$, is a procedure that when invoked returns a pair $(x, f(x))$ where $x \sim \{0,1\}^n$.

Let $0 < \varepsilon < 1$. We say that a circuit C with n inputs is *ε-close* to a function $f : \{0,1\}^n \to \{0,1\}$ if

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

We follow the intuitive notion of learning by Valiant [Val84].

Definition 2.22 (Infinitely-often learnability; following Valiant [Val84])**.** *A probabilistic algorithm* infinitely-often learns *a class of Boolean functions $\mathcal{F}$ with accuracy error ε and confidence error δ if, for all $f \in \mathcal{F}$ of the form $f = \{f_n\}_{n \in \mathbb{N}}$, whereby $f_n : \{0,1\}^n \to \{0,1\}$ for all $n \in \mathbb{N}$, and infinitely many $n \in \mathbb{N}$, it is the case that*

$$\Pr_{\mathrm{EX}(f_n), r}\left[A^{\mathrm{EX}(f_n)}(1^n; r) \text{ outputs a description of a circuit that is } \varepsilon(n)\text{-close to } f_n\right] \geq 1 - \delta(n)\,,$$

where the size of r is equal to the running time of A.

We will use the following result, which is inspired by Ilango, Loff, and Oliveira [ILO20].

Lemma 2.23 (Following Ilango, Loff, and Oliveira [ILO20, Theorem 36, Item (2)])**.** *If there exists $c > 1$ such that for all $0 < \gamma < 1$ $\mathrm{McKT}^m\mathrm{P}$ of dimension t and $m := t^{1+\gamma}$ can be solved on average with advantage $1/10$ for a size parameter $s := t^{1/c}$ and infinitely many $t \in \mathbb{N}$, then for every $a \geq 1$ the class $\mathsf{SIZE}[n^a]$ can be infinitely-often learned in polynomial time with accuracy error $1/n$ and confidence error $1/n$.*

Proof sketch. Let $a \geq 1$ be arbitrary, let $c > 1$ be as in the statement of Lemma 2.23, let $\gamma := 1/(c \cdot 4a)$, and let B be the PPT algorithm that witnesses the fact that $\mathrm{McKT}^m\mathrm{P}$ of dimension t and $m := t^{1+\gamma}$ can be solved on average with advantage $1/10$ for a size parameter $s := t^{1/c}$ and infinitely many $t \in \mathbb{N}$. Let $t \in \mathbb{N}$ be sufficiently large and such that B satisfies its average-case guarantees on inputs of dimension n. Let $n := t^\gamma$. Let $N := t + m(t) = t + t^{1+\gamma} = t + tn$ be the size of the instances of $\mathrm{McKT}^m\mathrm{P}$ of dimension t. Let q be a polynomial such that B runs in time $q(N)$. We will prove that $\mathsf{SIZE}[n^a]$ can be learned in polynomial time with accuracy error $1/n$ and confidence error $1/n$.

By examining the work of Ilango, Loff, and Oliveira [ILO20, Theorem 36, Item (2)], it would suffice to show that there exists a PPT algorithm B' such that for infinitely many $n \in \mathbb{N}$ and all $f \in \mathsf{SIZE}[n^a]$ it is the case that

$$\left| \Pr_{\{x_i\}_{i=1}^t, r}\left[B'(1^n, (x_1, f(x_1)), \dots, (x_t, f(x_t)); r) = 1\right] - \Pr_{\{x_i\}_{i=1}^t, b, r}\left[B'(1^n, (x_1, b_1), \dots, (x_t, b_t); r) = 1\right] \right| \\ \geq \frac{1}{10}$$

whereby $x_1, \dots, x_t \in \{0,1\}^n$, $b \in \{0,1\}^t$, and $r \in \{0,1\}^{\mathrm{poly}(N)}$ are independent and uniformly random. The reason is that Ilango, Loff, and Oliveira [ILO20] employ the inequality above in a hybrid argument that enables them to design a learning algorithm for $\mathsf{SIZE}[n^a]$, by making use of the advantage that the hybrid argument yields. Then, they use a boosting technique by Boneh and Lipton [BL93] to improve the accuracy of their learning algorithm.

To this end, let B' be a probabilistic algorithm such that

$$B'(1^n, z; r) := B(1^n, (z_{n+1}, z_{2n+2}, \dots, z_{t+tn}), \\ ((z_1, z_2, \dots, z_n), (z_{n+2}, z_{n+3}, \dots, z_{2n+1}), \dots, (z_{t+tn-n}, z_{t+tn-n+1}, \dots, z_{t+tn-1})); r),$$

for all $n \in \mathbb{N}$, $z \in \{0,1\}^{t+tn}$, and $r \in \{0,1\}^{q(N)}$. That is, on input $(1^n, z)$ and using random bits r the algorithm B' applies a permutation σ on z, to get a string $z' := \sigma(z) = z_{\sigma(1)} \cdots z_{\sigma(n)}$, and then runs B on $(1^n, z')$ using random bits r. For that matter B' runs in polynomial time, by the facts that σ is polynomial-time computable and B is a PPT algorithm.

We now have

$$\left| \Pr_{\{x_i\}_{i=1}^t, r}\left[B'(1^n, (x_1, f(x_1)), \dots, (x_t, f(x_t)); r) = 1\right] - \Pr_{\{x_i\}_{i=1}^t, b, r}\left[B'(1^n, (x_1, b_1), \dots, (x_t, b_t); r) = 1\right] \right| \\ = \left| \Pr_{\{x_i\}_{i=1}^t, r}\left[B(1^n, (f(x_1), \dots, f(x_t)), (x_1, \dots, x_t); r) = 1\right] - \Pr_{\{x_i\}_{i=1}^t, b, r}\left[B(1^n, b, (x_1, \dots, x_t); r) = 1\right] \right| \\ \geq \frac{1}{10}$$

by the definition of B', our assumption, and the fact that $\mathrm{KT}((f(x_1), \dots, f(x_t)) \mid (x_1, \dots, x_t)) \leq O\left((n^a)^3\right) \leq n^{4a} = n^{\gamma/c} = t^{1/c} = s < t/4$, by Lemma 2.12. □

2.11 Pseudorandom generators

We recount the notion of fooling a PPT algorithm.

Definition 2.24. *Let $\ell, n \in \mathbb{N}$ be such that $\ell < n$, and $G : \{0,1\}^\ell \to \{0,1\}^n$ be a function. Let A be a PPT algorithm that on inputs of size $k \in \mathbb{N}$ runs in time $q(k)$ for some polynomial q. Let also $0 < \varepsilon < 1$.*

We say that G is a function that ε-fools A, *if*

$$\left| \mathop{\mathbf{Pr}}_{\substack{x\sim\{0,1\}^n,\\ r\sim\{0,1\}^{q(n)}}} [A(x;r)=1] - \mathop{\mathbf{Pr}}_{\substack{y\sim\{0,1\}^\ell,\\ r\sim\{0,1\}^{q(n)}}} [A(G(y)\,;r)=1] \right| < \varepsilon.$$

The notion of fooling is used in the definition of a *pseudorandom generator (PRG)*. We will make use of PRGs $\{G_n\}_{n\in\mathbb{N}}$ for which there exists a function $\ell : \mathbb{N}\to\mathbb{N}$ that satisfies $\ell(n) < n$ for all $n\in\mathbb{N}$, such that $G_n : \{0,1\}^{\ell(n)} \to \{0,1\}^n$ for all $n\in\mathbb{N}$ and the following holds: For every PPT algorithm A there exists a function $\varepsilon : \mathbb{N}\to[0,1]$ such that for all $n\in\mathbb{N}$ it is the case that G_n is a function that $\varepsilon(n)$-fools A.

Håstad, Impagliazzo, Levin, Luby [HILL99] have shown that the existence of OWFs implies the existence of PRGs.

Theorem 2.25 (OWFs imply PRGs; see Håstad, Impagliazzo, Levin, Luby [HILL99])**.** *If there exists a OWF, then for every* $c>0$ *there exists a function* $\{G_n\}_{n\in\mathbb{N}}$*, whereby* $G_n : \{0,1\}^{n^{1/c}} \to \{0,1\}^n$ *for all* $n\in\mathbb{N}$*, such that for every PPT algorithm* A *there is a negligible function* $\varepsilon : \mathbb{N}\to[0,1]$ *such that for all* $n\in\mathbb{N}$ *it is the case that* G_n *is a function that* $\varepsilon(n)$*-fools* A*.*

2.12 Pseudorandom functions

We will also require the notions of pseudorandom function families and distinguishers for function families.

Definition 2.26. *Let* $\{f_y\}_{y\in\{0,1\}^*}$ *be a family of functions such that* $f_y : \{0,1\}^{|y|} \to \{0,1\}$ *for all* $y\in\{0,1\}^*$*, and there is a polynomial-time algorithm that computes* $f_y(x)$ *given* y *and* $x\in\{0,1\}^{|y|}$*. We say that the function family* $\{f_y\}_{y\in\{0,1\}^*}$ *is a* pseudorandom function family (PRF family) *if for all PPT oracle algorithms* A *there is a negligible function* $\varepsilon : \mathbb{N}\to[0,1]$ *such that for all sufficiently large* $n\in\mathbb{N}$ *it is the case that*

$$\left| \mathop{\mathbf{Pr}}_{y\sim\{0,1\}^n,r}\left[A^{f_y}(1^n;r)=1\right] - \mathop{\mathbf{Pr}}_{g\sim\mathcal{F}_n,r}\left[A^{g}(1^n;r)=1\right] \right| < \varepsilon(n)$$

where the size of r *is equal to the running time of* A*.*

Definition 2.27. *Let* $\{f_y\}_{y\in\{0,1\}^*}$ *be a family of functions such that* $f_y : \{0,1\}^{|y|} \to \{0,1\}$ *for all* $y\in\{0,1\}^*$*, and there is a polynomial-time algorithm that computes* $f_y(x)$ *given* y *and* $x\in\{0,1\}^{|y|}$*. We say that a PPT oracle algorithm* A *is a* distinguisher for $\{f_y\}_{y\in\{0,1\}^*}$ *if for all negligible functions* $\varepsilon : \mathbb{N}\to[0,1]$ *and infinitely many* $n\in\mathbb{N}$ *it is the case that*

$$\left| \mathop{\mathbf{Pr}}_{y\sim\{0,1\}^n,r}\left[A^{f_y}(1^n;r)=1\right] - \mathop{\mathbf{Pr}}_{g\sim\mathcal{F}_n,r}\left[A^{g}(1^n;r)=1\right] \right| \geq \varepsilon(n)$$

where the size of r *is equal to the running time of* A*.*

Note how a distinguisher violates the guarantees of a PRF family. Below, we present a result by Goldreich, Goldwasser, and Micali [GGM86], where they proved that the existence of PRGs implies the existence of PRFs.

Theorem 2.28 (PRGs imply PRFs; see Goldreich, Goldwasser, and Micali [GGM86])**.** *If there exists a function* $\{G_n\}_{n\in\mathbb{N}}$*, whereby* $G_n : \{0,1\}^{n/2} \to \{0,1\}^n$ *for all* $n\in\mathbb{N}$*, such that for every PPT algorithm* A *there is a negligible function* $\varepsilon : \mathbb{N}\to[0,1]$ *such that for all* $n\in\mathbb{N}$ *it is the case that* G_n *is a function that* $\varepsilon(n)$*-fools* A*, then there exists a PRF family.*

Theorem 2.25 and Theorem 2.28 yield the following corollary.

Corollary 2.29 (OWFs imply PRFs)**.** *If there exists a OWF, then there exists a PRF family.*

3 OWFs from average-case hardness of McKTP

In this section, we prove the following result.

Theorem 3.1. *Assume that, for some $m : \mathbb{N} \to \mathbb{N}$, $\mathrm{McKT}^m\mathrm{P}$ of dimension n is $(1/p)$-HoA for some polynomial p. Then, there exists some weak OWF.*

By Theorem 3.1 and Theorem 2.18, we get the following corollary.

Corollary 3.2 (Theorem 1.1, restated)**.** *Assume that, for some $m : \mathbb{N} \to \mathbb{N}$, $\mathrm{McKT}^m\mathrm{P}$ of dimension n is $(1/p)$-HoA for some polynomial p. Then, there exists some OWF.*

3.1 Proof of Theorem 3.1

We will first require the following two lemmas.

Lemma 3.3. *For all functions $m : \mathbb{N} \to \mathbb{N}$, if $\mathrm{McKT}^m\mathrm{P}$ is $(1/p)$-HoA for some polynomial p, then* Search $\mathrm{McKT}^m\mathrm{P}$ *is $\left(1/p^2\right)$-HoA.*

Proof. We will prove the contrapositive. That is, we will prove that if Search $\mathrm{McKT}^m\mathrm{P}$ is $\left(1-1/p^2\right)$-EoA, then $\mathrm{McKT}^m\mathrm{P}$ is $(1-1/p)$-EoA. In what follows, let $c > 0$ be as in Corollary 2.9.

Let $N' := n + m(n)$ be the size of the instances of Search $\mathrm{McKT}^m\mathrm{P}$ of dimension n. Assume that Search $\mathrm{McKT}^m\mathrm{P}$ is $\left(1-1/p^2\right)$-EoA. That is, assume that there exists some heuristic H' that on input a random instance $(x, y) \in \{0,1\}^n \times \{0,1\}^{m(n)}$ outputs with probability greater than $1 - 1/p(N')^2$ a program $\Pi \in \{0,1\}^*$ and a run-time bound $t \in \mathbb{N}$ (in binary) such that $U^{\Pi,y}(i, 1^t) = x_i$ for all $1 \le i \le n$, and the sum $|\Pi| + t$ is minimized over the choices of Π and t.

Given H', a heuristic H for $\mathrm{McKT}^m\mathrm{P}$ of dimension n and input size $N := n + m(n) + \log(n + c\log n)$, works as follows:

> On input strings $x \in \{0,1\}^n$ and $y \in \{0,1\}^{m(n)}$, and a size parameter $0 \le \theta \le n + c\log n$ in binary, run H' on (x, y) to get a program $\Pi \in \{0,1\}^*$ and a run-time bound $t \in \mathbb{N}$ (in binary). If Π and t are such that $U^{\Pi,y}(i, 1^t) = x_i$ for all $1 \le i \le n$ and $|\Pi| + t \le \theta$, then return YES. Else, return NO.

Note that the running time of H is polynomial in N. The success probability of H over a random instance (x, y, θ) and random bits r is

$$\begin{aligned}
&\Pr_{x,y,\theta,r}[H(x,y,\theta;r) \text{ succeeds}] \\
&\quad \ge \Pr_{x,y,\theta,r}[H(x,y,\theta;r) \text{ succeeds} \mid H'(x,y;r) \text{ succeeds}] \cdot \Pr_{x,y,r}[H'(x,y;r) \text{ succeeds}] \\
&\quad > 1 \cdot \left(1 - \frac{1}{p(N')^2}\right) \\
&\quad = 1 - \frac{1}{p(N')^2} \\
&\quad \ge 1 - \frac{1}{p(N)},
\end{aligned}$$

since $1/p(N')^2 \le 1/p(N)$ for all sufficiently large $n \in \mathbb{N}$, as desired.

Therefore, $\mathrm{McKT}^m\mathrm{P}$ is $(1-1/p)$-EoA as witnessed by H. □

The following is an elaboration on the seminal work by Liu and Pass [LP20].

Lemma 3.4 (Following Liu and Pass [LP20])**.** *Assume that, for some $m : \mathbb{N} \to \mathbb{N}$,* Search $\mathrm{McKT}^m\mathrm{P}$ *is $(1/p)$-HoA for some polynomial p. Then, there exists some weak OWF.*

Proof. Fix some UTM U, and let $c > 0$ be as in Corollary 2.9. Let $n \in \mathbb{N}$ be sufficiently large and such that Search $\mathrm{McKT}^m\mathrm{P}$ of dimension n is $(1/p)$-HoA. Consider the function $f : \{0,1\}^* \to \{0,1\}^*$ defined by the mapping rule

$$(\ell, t, y, \Pi') \mapsto \left(\ell + t, U^{\Pi,y}\left(1, 1^t\right), \ldots, U^{\Pi,y}\left(n, 1^t\right), y\right),$$

where $m := m(n)$, $y \in \{0,1\}^m$, $\Pi' \in \{0,1\}^{n+c\log n}$ is a program, and $\Pi := \Pi'|_{[\ell]}$ is the ℓ-bit prefix of Π'. Note that without loss of generality, $\ell + t \leq n + c\log n$, by Corollary 2.9. This also implies that $\ell \leq n + c\log n$ and $t \leq n + c\log n$. For that matter, f is a function from $\{0,1\}^M$ to $\{0,1\}^N$, where $M := \log(2n + 2c\log n) + m + n + c\log n$ and $N := \log(n + c\log n) + n + m$, and is computable in polynomial time.

Observe also that f is only defined over infinitely many input lengths. However, by a padding trick, f can be transformed into another function f' that is defined over all input lengths, and such that f' is a weak one-way function, given that f is [LP20].

We now claim that if Search McKT^{m}P is $(1/p)$-HoA, then f is a $(1/q)$-weak OWF, where q is a polynomial such that $q(n) := 2\,(2n + 2c\log n)\, n^c p(n + m(n))^3$ for all $n \in \mathbb{N}$. Towards a contradiction, assume that there exists a PPT algorithm A that inverts f with probability at least $1 - 1/q(M) \geq 1 - 1/q(n)$.

First, note that except for a fraction $1/\left(2p(n+m)\right)$ of sequences of random bits r for A, the deterministic machine A_r, given by $A_r(f(z)) := A(f(z)\,;r)$ for all $z \in \{0,1\}^M$, fails to invert f with probability at most $2p(n+m)/q(n)$ over a uniformly random input z. This is so, as

$$\begin{aligned}
&\mathop{\mathbf{Pr}}_{r}\left[\mathop{\mathbf{Pr}}_{z}[A_r(f(z)) \text{ fails}] > \frac{2p(n+m)}{q(n)}\right] \\
&\quad\leq \mathop{\mathbf{Pr}}_{r}\left[\mathop{\mathbf{Pr}}_{z}[A_r(f(z)) \text{ fails}] \geq 2p(n+m) \cdot \mathop{\mathbf{Pr}}_{z,r}[A_r(f(z)) \text{ fails}]\right] \\
&\quad= \mathop{\mathbf{Pr}}_{r}\left[\mathop{\mathbf{Pr}}_{z}[A(f(z)\,;r) \text{ fails}] \geq 2p(n+m) \cdot \mathop{\mathbf{E}}_{r}\left[\mathop{\mathbf{Pr}}_{z}[A(f(z)\,;r) \text{ fails}]\right]\right] \\
&\quad\leq \frac{1}{2p(n+m)},
\end{aligned}$$

by Lemma 2.2. Henceforth, we will call such a sequence of random bits *good*; otherwise, we will call a sequence of random bits *bad*. Therefore, we have

$$\mathop{\mathbf{Pr}}_{z,r}[A(f(z)\,;r) \text{ fails} \mid r \text{ is good}] = \mathop{\mathbf{Pr}}_{z,r}[A_r(f(z)) \text{ fails} \mid r \text{ is good}] \leq \frac{2p(n+m)}{q(n)}.$$

We propose the following heuristic H for Search McKT^{m}P:

> On input strings $x \in \{0,1\}^m$ and $y \in \{0,1\}^m$, and using random bits r, the algorithm H runs $A(j, x, y; r)$ for all $j \in [n + c\log n]$. For each $j \in [n + c\log n]$, $A(j, x, y; r)$ returns a tuple (s_j, t_j, y, Π'_j). Then, $H(x, y; r)$ returns a program from $\left\{ \Pi'_j|_{[s_j]} \right\}_{j \in [n + c\log n]}$ such that
>
> $$U^y\left(\Pi'_j|_{[s_j]}, i, 1^{t_j}\right) = x_i$$
>
> for all $1 \leq i \leq n$, and $\left|\Pi'_j|_{[s_j]}\right| + t_j = s_j + t_j$ is minimized.

We will now analyze the average-case performance of H. Fix a good sequence of random bits r, as defined above, and recall that, in this case, $\mathbf{Pr}_z[A_r(f(z)) \text{ fails}] \leq 2p(n+m)/q(n)$. Let S_r be the set of inputs (x, y) for which $H(x, y; r)$ fails, when given random bits r. Observe that, for any good r,

$$\mathop{\mathbf{Pr}}_{x,y}[H(x, y; r) \text{ fails}] = \frac{|S_r|}{2^{n+m}}.$$

Consider $(x, y) \in S_r$ and let $w_{x,y} := \mathrm{KT}(x \mid y)$ be the conditional KT-complexity of x given y. By Corollary 2.9, we have $w_{x,y} \leq n + c\log n$. If $H(x, y; r)$ fails, then it means that A fails to invert $(w_{x,y}, x, y)$ when given the good sequence of random bits r.

Recall that $\mathbf{Pr}_z[A_r(f(z)) \text{ fails}] \leq 2p(m\,(n+1))/q(n)$. Recall also, from the definition of f, and from the fact that $w_{x,y} \leq n + c\log n$, that

$$\mathop{\mathbf{Pr}}_{z}[f(z) = (w_{x,y}, x, y)] \geq \frac{1}{(2n + 2c\log n) \cdot 2^m \cdot 2^{n+c\log n}}.$$

Thus, for any good sequence r, we have

$$\begin{aligned}
\frac{2p(n+m)}{q(n)} &\geq \Pr_{z}[A_r\,(f((z))\text{ fails}] \\
&= \sum_{(w,x,y):A_r(w,\,x,\,y)\text{ fails}} \Pr_{z}[f(z) = (w,x,y)] \\
&\geq \sum_{(x,y):A_r(w_{x,y},\,x,\,y)\text{ fails}} \Pr_{z}[f(z) = (w_{x,y},x,y)] \\
&\geq \sum_{(x,y)\in S_r} \frac{1}{(2n+2c\log n)\cdot 2^m \cdot 2^{n+c\log n}} \\
&= \frac{|S_r|}{2^{n+m}} \cdot \frac{1}{(2n+2c\log n)\,2^{c\log n}} \\
&= \frac{\Pr_{x,y}[H(x,y;r)\text{ fails}]}{(2n+2c\log n)\,n^c}.
\end{aligned}$$

Since this holds for any good sequence r, we have that

$$\begin{aligned}
\Pr_{x,y,r}[H(x,y;r)\text{ fails} \mid r\text{ is good}] &\leq \frac{(2n+2c\log n)\,n^c 2p(n+m)}{q(n)} \\
&= \frac{(2n+2c\log n)\,n^c 2p(n+m)}{2\,(2n+2c\log n)\,n^c p(n+m)^3} \\
&= \frac{1}{p(n+m)^2} \\
&< \frac{1}{2p(n+m)},
\end{aligned}$$

since $p(n+m) > 2$ for all sufficiently large $n \in \mathbb{N}$. Therefore, H fails with probability at most

$$\Pr_{x,y,r}[H(x,y;r)\text{ fails} \mid r\text{ is good}] + \Pr_{r}[r\text{ is bad}] < \frac{1}{2p(n+m)} + \frac{1}{2p(n+m)} = \frac{1}{p(n+m)}.$$

This yields a contradiction. □

We now turn to the proof of Theorem 3.1.

Proof of Theorem 3.1. Immediate; by Lemma 3.3 and Lemma 3.4, since if p is a polynomial, then p^2 is a polynomial too. □

4 Average-case hardness of McKTP from OWFs

In this section, we prove the following result, which is a *partial* converse to Theorem 3.1.

Theorem 4.1 (Theorem 1.3, restated)**.** *Assume that there exists a OWF. Then, there exists a function $m : \mathbb{N} \to \mathbb{N}$ such that* $\mathrm{McKT}^m\mathrm{P}$ *of dimension n is $(1/h)$-HoA, for a function $h : \mathbb{N} \to \mathbb{R}_{>0}$ such that $h(N) := 2^{N/100}$ for all $N \in \mathbb{N}$.*

4.1 Applications of heuristics

We will require the following result, which asserts that heuristics with good average-case performance guarantees can be used to design learning algorithms.

Lemma 4.2. *If for all $0 < \gamma < 1$ it is the case that* $\mathrm{McKT}^m\mathrm{P}$ *of dimension n and $m := n^{1+\gamma}$ is $(1-1/h)$-EoA, for a function $h : \mathbb{N} \to \mathbb{R}_{>0}$ such that $h(N) := 2^{N/100}$ for all $N \in \mathbb{N}$, then for every $a \geq 1$ the class* $\mathsf{SIZE}[n^a]$ *can be infinitely-often learned in polynomial time with accuracy error $\varepsilon = 1/n$ and confidence error $\delta = 1/n$.*

Proof. We will apply Lemma 2.23, by proving that if for all $0 < \gamma < 1$ it is the case that McKT^{m}P of dimension n and $m := n^{1+\gamma}$ is $(1 - 1/h)$-EoA, then for all $c > 1$ and all $0 < \gamma < 1$ it is the case that McKT^{m}P of dimension n and $m := n^{1+\gamma}$ is EoA with advantage $1/10$ for a size parameter $s := n^{1/c}$ and infinitely many $n \in \mathbb{N}$. The desired result will then follow from Lemma 2.23.

Let $0 < \gamma < 1$ be arbitrary. Let H be the heuristic that witnesses the fact that McKT^{m}P of dimension n and $m := n^{1+\gamma}$ is $(1 - 1/h)$-EoA (see Definition 2.20), and assume that H runs in time $q(N)$ for some polynomial q.

Let $n \in \mathbb{N}$ be sufficiently large and such that H satisfies its average-case performance guarantees on inputs of dimension n. In what follows, let $N := n + m(n) + \log(n + \sigma \log n)$ be the size of the McKT^{m}P on instances of dimension n, where σ is from Corollary 2.9; see Definition 2.13.

Let $c > 1$ be arbitrary, and $s := n^{1/c}$. Let H^* be a probabilistic algorithm such that, for all strings $x \in \{0,1\}^n$ and $y \in \{0,1\}^{m(n)}$, and all random strings $r \in \{0,1\}^{q(N)}$, satisfies $H^*(1^n, x, y; r) := H(x, y, s; r)$.

By the definition of H^*, we have that the first term of the LHS of the inequality of Definition 2.21 is

$$\Pr_{y,r}[H^*(1^n, x, y; r) = 1] = \Pr_{y,r}[H(x, y, s; r) = 1]\,,$$

where $y \in \{0,1\}^{m(n)}$ is uniformly random, and $x = x(y) \in \{0,1\}^n$ is such that $\mathrm{KT}(x \mid y) \leq s$, as guaranteed to exist by Lemma 2.10. We claim that this probability is sufficiently large.

Claim 4.3. *It is the case that*

$$\Pr_{y,r}[H(x, y, s; r) = 1] \geq \frac{1}{5}.$$

Proof. Towards a contradiction, assume that

$$\Pr_{y,r}[H(x, y, s; r) = 1] < \frac{1}{5}.$$

Then, the number of inputs and random bits of H that make H output "0" is greater than

$$K := 2^{m(n)} \cdot 2^{q(N)} \cdot \left(1 - \frac{1}{5}\right) = 2^{m(n)+q(N)} \cdot \frac{4}{5}.$$

However, it is the case that $K < 2^{N+q(N)}/h(N)$, by our assumption on the average-case performance of H. So

$$2^{m(n)+q(N)} \cdot \frac{4}{5} < \frac{1}{h(N)} \cdot 2^{N+q(N)} = \frac{1}{2^{N/100}} \cdot 2^{n+m(n)+\log(n+c\log n)+q(N)}$$

or

$$\frac{4}{5} < \frac{1}{2^{N/100}} \cdot 2^{n+\log(n+c\log n)} \leq \frac{1}{2^{n/100}} \cdot 2^{2n} < \frac{4}{500};$$

this yields a contradiction. (Claim 4.3) □

We now turn to the second term of the LHS of the inequality of Definition 2.21. To this end, we have

$$\begin{aligned}
\Pr_{z,y,r}[H^*(1^n, z, y; r)] &= \Pr_{z,y,r}[H(z, y, s; r) = 1] \\
&\leq \Pr_{z,y,r}[H(z, y, s; r) = 1 \mid \mathrm{KT}(z \mid y) > s] + \Pr_{z,y}[\mathrm{KT}(z \mid y) \leq s] \\
&= \frac{\Pr_{z,y,r}[H(z, y, s; r) = 1 \text{ and } \mathrm{KT}(z \mid y) > s]}{\Pr_{z,y}[\mathrm{KT}(z \mid y) > s]} + \Pr_{z,y}[\mathrm{KT}(z \mid y) \leq s] \\
&\leq \frac{1/h(N)}{1/2} + \frac{1}{20},
\end{aligned}$$

by Lemma 2.11, or

$$\begin{aligned}
&= \frac{2}{h(N)} + \frac{1}{20} \\
&= \frac{2}{2^{N/100}} + \frac{1}{20}
\end{aligned}$$

$$
\begin{aligned}
&\leq \frac{2}{2^{n/100}} + \frac{1}{20}\\
&\leq \frac{1}{20} + \frac{1}{20}\\
&= \frac{1}{10},
\end{aligned}
$$

Therefore, by Claim 4.3 and the discussion above,

$$\mathop{\mathbf{Pr}}_{y,r}[H^*(1^n, x, y; r) = 1] - \mathop{\mathbf{Pr}}_{z,y,r}[H^*(1^n, z, y; r)] \geq \frac{1}{5} - \frac{1}{10} = \frac{1}{10},$$

as desired. □

Remark 4.4. *It should be noted that Lemma 4.2 also holds for the case where* McKTP *is* zero-error *easy on average.*

4.2 Applications of infinitely-often learning algorithms

An important observation is that a learning algorithm for polynomial-size circuits may be used to create distinguishers for polynomial-time computable function families.

Lemma 4.5 (See also Oliveira and Santhanam [OS17, Theorem 8])**.** *Assume that for every $a \geq 1$ the class $\mathsf{SIZE}[n^a]$ can be infinitely-often learned in polynomial time with accuracy error $\varepsilon = 1/n$ and confidence error $\delta = 1/n$. Then, for all function families $\{f_y\}_{y \in \{0,1\}^*}$ such that $f_y : \{0,1\}^{|y|} \to \{0,1\}$ for all $y \in \{0,1\}^*$, and there is a polynomial-time algorithm that computes $f_y(x)$ given y and $x \in \{0,1\}^{|y|}$, there is a distinguisher for $\{f_y\}_{y \in \{0,1\}^*}$.*

Proof. Let $\{f_y\}_{y \in \{0,1\}^*}$ be a function family such that $f_y : \{0,1\}^{|y|} \to \{0,1\}$ for all $y \in \{0,1\}^*$, and there is a polynomial-time algorithm that computes $f_y(x)$ given y and $x \in \{0,1\}^{|y|}$. In particular, for all $y \in \{0,1\}^*$ it is the case that $f_y : \{0,1\}^{|y|} \to \{0,1\}$ is computable in time $|y|^{k/2}$ for some $k > 0$. By Lemma 2.7, for all $y \in \{0,1\}^*$ it is the case that $f_y : \{0,1\}^{|y|} \to \{0,1\}$ is computable by some circuit of size

$$O\Big(|y|^{k/2} \log |y|\Big) \leq |y|^k .$$

Let A be the PPT learning algorithm for $\mathsf{SIZE}\big[n^k\big]$ that works for infinitely many $n \in \mathbb{N}$, and that runs in time $q(n)$ for some polynomial q, as guaranteed to exist by our assumption. In particular, let $n \in \mathbb{N}$ be sufficiently large and such that A satisfies its learning guarantees on inputs of size n. Let $t := n^{100}$ and define D to be a probabilistic oracle algorithm as follows.

> On input 1^n, random bits $r := (r', r'') \in \{0,1\}^{O(q(n))} \times \{0,1\}^{t \cdot n}$, and given oracle access to some function $g : \{0,1\}^n \to \{0,1\}$, the algorithm D runs A on 1^n using random bits $r' \in \{0,1\}^{O(q(n))}$ to get a hypothesis h for g, whereby simulating calls to $\mathrm{EX}(g)$ by using the oracle for g. Then, D samples t strings $x_1, \ldots, x_t$ from $\{0,1\}^n$ using random bits $r'' \in \{0,1\}^{t \cdot n}$ and uses the oracle for g to compute
>
> $$\alpha := \frac{|\{i \in [t] \mid h(x_i) = g(x_i)\}|}{t}.$$
>
> Finally, if $\alpha \geq 2/3$, then D outputs 1; else, D outputs 0.

Note that D runs in time polynomial in n. We will now prove that D is a distinguisher for $\{f_y\}_{y \in \{0,1\}^*}$. To this end, we will show that D satisfies Definition 2.27.

The first term of the LHS of the inequality of Definition 2.27 is

$$
\begin{aligned}
\mathop{\mathbf{Pr}}_{y \sim \{0,1\}^n, r}\big[D^{f_y}(1^n; r) = 1\big] &= \mathop{\mathbf{Pr}}_{y,r}\left[\alpha \geq \frac{2}{3}\right]\\
&= \mathop{\mathbf{Pr}}_{y,r}\left[\frac{|\{i \in [t] \mid h(x_i) = f_y(x_i)\}|}{t} \geq \frac{2}{3}\right]
\end{aligned}
$$

$$\begin{aligned}
&\geq \Pr_{y,r}\left[\frac{|\{i\in[t] \mid h(x_i)=f_y(x_i)\}|}{t} \geq \frac{3}{4}\left(1-\frac{1}{n}\right)\right] \\
&= \Pr_{y,r}\left[\frac{|\{i\in[t] \mid h(x_i)=f_y(x_i)\}|}{t} \geq \frac{3}{4}(1-\varepsilon)\right] \\
&\geq \Pr_{y,r}\left[\frac{|\{i\in[t] \mid h(x_i)=f_y(x_i)\}|}{t} \geq \frac{3}{4}(1-\varepsilon) \mid h \text{ is } \varepsilon\text{-close to } f_y\right] \\
&\qquad \cdot \Pr_{y,r}[h \text{ is } \varepsilon\text{-close to } f_y]\,.
\end{aligned}$$

For all $1 \leq i \leq t$, let X_i be a Boolean variable such that $X_i := 1$ if and only if $h(x_i) = f_y(x_i)$. Then,

$$\begin{aligned}
\Pr_{y,r}\big[D^{f_y}(1^n;r)=1\big] &\geq \Pr_{y,r}\left[\frac{\sum_{i=1}^t X_i}{t} \geq \frac{3}{4}(1-\varepsilon) \mid h \text{ is } \varepsilon\text{-close to } f_y\right] \cdot \Pr_{y,r}[h \text{ is } \varepsilon\text{-close to } f_y] \\
&\geq \Pr_{y,r}\left[\frac{\sum_{i=1}^t X_i}{t} \geq \frac{3}{4}\mathop{\mathbf{E}}_{y,r}\left[\frac{\sum_{i=1}^t X_i}{t}\right]\right](1-\delta)\,,
\end{aligned}$$

as $\mathrm{CC}(f_y) \leq |y|^k = n^k$, or

$$\geq \frac{1}{4}\mathop{\mathbf{E}}_{y,r}\left[\frac{\sum_{i=1}^t X_i}{t}\right](1-\delta)\,,$$

by Lemma 2.1, or

$$\begin{aligned}
&\geq \frac{1}{4}(1-\varepsilon)(1-\delta) \\
&= \frac{1}{4}\left(1-\frac{1}{n}\right)\left(1-\frac{1}{n}\right) \\
&\geq \frac{1}{4}-\frac{1}{8} \\
&= \frac{1}{8}.
\end{aligned}$$

We now turn to the second term of the LHS of the inequality of Definition 2.27. To this end, we have

$$\begin{aligned}
\Pr_{g\sim\mathcal{F}_n,r}[D^g(1^n;r)=1] &= \Pr_{g,r}\left[\alpha \geq \frac{2}{3}\right] \\
&= \Pr_{g,r}\left[\frac{|\{i\in[t] \mid h(x_i)=g(x_i)\}|}{t} \geq \frac{2}{3}\right] \\
&\leq \Pr_{g,r}\left[\frac{|\{i\in[t] \mid h(x_i)=g(x_i)\}|}{t} \geq \frac{2}{3} \mid \{x_i\}_{i=1}^t \text{ are distinct}\right] \\
&\qquad + \Pr\Big[\{x_i\}_{i=1}^t \text{ are not distinct}\Big] \\
&= \Pr_{b,r}\left[|\{i\in[t] \mid h(x_i)=b_i\}| \geq \frac{2t}{3} \mid \{x_i\}_{i=1}^t \text{ are distinct}\right] \\
&\qquad + \Pr\Big[\{x_i\}_{i=1}^t \text{ are not distinct}\Big]\,,
\end{aligned}$$

where $b_1, \ldots, b_t \in \{0,1\}$ are independent and uniformly random, and $b := (b_1, \ldots, b_t)$.

Similarly as above, for all $1 \leq i \leq t$, let X_i be a Boolean variable such that $X_i := 1$ if and only if $h(x_i) = b_i$. Let Y be the event that all of the x_i are distinct. Then,

$$\Pr_{g\sim\mathcal{F}_n,r}[D^g(1^n;r)=1] \leq \Pr_{b,r}\left[\sum_{i=1}^t X_i \geq \frac{2t}{3} \mid Y\right] + \Pr\Big[\{x_i\}_{i=1}^t \text{ are not distinct}\Big]$$

$$
\begin{aligned}
&= \Pr_{b,r}\left[\sum_{i=1}^{t} X_i \geq \frac{4}{3}\cdot\frac{t}{2} \mid Y\right] + \mathbf{Pr}[\exists i,j \in [t] : i \neq j \text{ and } x_i = x_j] \\
&\leq \Pr_{b,r}\left[\sum_{i=1}^{t} X_i \geq \frac{4}{3}\cdot\frac{t}{2} \mid Y\right] + \binom{t}{2}2^{-n},
\end{aligned}
$$

by a union bound, or

$$
\begin{aligned}
&= \Pr_{b,r}\left[\sum_{i=1}^{t} X_i \geq \left(1+\frac{1}{3}\right)\mathop{\mathbf{E}}_{b,r}\left[\sum_{i=1}^{t} X_i \mid Y\right] \mid Y\right] + \binom{n^{100}}{2}2^{-n} \\
&\leq \left(e^{-\frac{1/9}{3}}\right)^{\mathbf{E}_{b,r}\left[\sum_{i=1}^{t} X_i | Y\right]} + \frac{n^{200}}{2^n},
\end{aligned}
$$

by Lemma 2.3, or

$$
\begin{aligned}
&\leq \left(e^{-1/27}\right)^{t/2} + \frac{1}{32} \\
&= e^{-t/54} + \frac{1}{32} \\
&= e^{-n^{100}/54} + \frac{1}{32} \\
&\leq \frac{1}{32} + \frac{1}{32} \\
&= \frac{1}{16}.
\end{aligned}
$$

Therefore,

$$
\left| \Pr_{y\sim\{0,1\}^n, r}\left[D^{f_y}(1^n; r) = 1\right] - \Pr_{g\sim\mathcal{F}_n, r}\left[D^{g}(1^n; r) = 1\right] \right| \geq \frac{1}{8} - \frac{1}{16} = \frac{1}{16} = \Omega(1)
$$

and as every negligible function $\mu : \mathbb{N} \to [0,1]$ is such that $\mu(n) = o(1) < \Omega(1)$, the desired result follows. □

4.3 Proof of Theorem 4.1

We will prove the contrapositive. To this end, assume that for all functions $m : \mathbb{N} \to \mathbb{N}$ it is the case that McKT^{m}P of dimension n is $(1-1/h)$-EoA, for a function $h : \mathbb{N} \to \mathbb{R}_{>0}$ such that $h(N) := 2^{N/100}$ for all $N \in \mathbb{N}$. This implies that for all $0 < \gamma < 1$ it is the case that McKT^{m}P of dimension n and $m := n^{1+\gamma}$ is $(1-1/h)$-EoA, for a function $h : \mathbb{N} \to \mathbb{R}_{>0}$ such that $h(N) := 2^{N/100}$ for all $N \in \mathbb{N}$.

By Lemma 4.2, for all $a \geq 1$, we get a learning algorithm for $\mathsf{SIZE}[n^a]$ that works for infinitely many $n \in \mathbb{N}$. By Lemma 4.5, for every function family $\{f_y\}_{y\in\{0,1\}^*}$ such that $f_y : \{0,1\}^{|y|} \to \{0,1\}$ for all $y \in \{0,1\}^*$, and there is a polynomial-time algorithm that computes $f_y(x)$ given y and $x \in \{0,1\}^{|y|}$, there is a distinguisher for $\{f_y\}_{y\in\{0,1\}^*}$. By Corollary 2.29, there are no OWFs.

Acknowledgements

We would like to thank Russell Impagliazzo for explaining his work [IL90] to us, and Ján Pich and Ninad Rajgopal for illuminating discussions. We thank Ján Pich for bringing his work [Pic20] to our attention. We thank Mikito Nanashima and Hanlin Ren for helpful comments and for spotting bugs in the proofs of earlier versions of Lemma 3.3 and Lemma 3.4, respectively. In particular, we thank Hanlin Ren for asking the question of whether KT complexity would be an appropriate complexity measure to consider in the context of our work.

This work was partly carried out during a visit of Dimitrios Myrisiotis to DIMACS, with support from the Special Focus on Lower Bounds in Computational Complexity program.

Eric Allender and Harsha Tirumala were partially supported by NSF Grant CCF-1909216; Harsha Tirumala was also partially supported by the Simons Collaboration on Algorithms and Geometry. Mahdi Cheraghchi's research was partially supported by the National Science Foundation under Grant No. CCF-2006455.

References

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

[AGGM06] Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. On basing one-way functions on NP-hardness. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 701–710. ACM, 2006. See also [AGGM10].

[AGGM10] Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. Erratum for: On basing one-way functions on NP-hardness. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC)*, pages 795–796. ACM, 2010.

[BC20] Chris Brzuska and Geoffroy Couteau. Towards fine-grained one-way functions from strong average-case hardness. *IACR Cryptol. ePrint Arch.*, 2020:1326, 2020.

[BL93] Dan Boneh and Richard J. Lipton. Amplification of weak learning under the uniform distribution. In Lenny Pitt, editor, *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory, COLT 1993, Santa Cruz, CA, USA, July 26-28, 1993*, pages 347–351. ACM, 1993.

[BT06a] Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Found. Trends Theor. Comput. Sci.*, 2(1), 2006.

[BT06b] Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for NP problems. *SIAM J. Comput.*, 36(4):1119–1159, 2006.

[DS14] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In David B. Shmoys, editor, *Symposium on Theory of Computing (STOC)*, pages 624–633. ACM, 2014.

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

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

[Hir18] Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *59th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 247–258. IEEE Computer Society, 2018.

[HS17] Shuichi Hirahara and Rahul Santhanam. On the average-case complexity of MCSP and its variants. In Ryan O'Donnell, editor, *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, volume 79 of *LIPIcs*, pages 7:1–7:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[IL90] Russell Impagliazzo and Leonid A. Levin. No better ways to generate hard NP instances than picking uniformly at random. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, pages 812–821. IEEE Computer Society, 1990.

[Ila20] Rahul Ilango. Approaching MCSP from above and below: Hardness for a conditional variant and $\mathsf{AC}^0[p]$. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 34:1–34:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[ILO20] Rahul Ilango, Bruno Loff, and Igor Carboni Oliveira. NP-hardness of circuit minimization for multi-output functions. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 22:1–22:36. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[Imp95] Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference, Minneapolis, Minnesota, USA, June 19-22, 1995*, pages 134–147. IEEE Computer Society, 1995.

[LP20] Yanyi Liu and Rafael Pass. On one-way functions and Kolmogorov complexity. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1243–1254. IEEE, 2020.

[LP21a] Yanyi Liu and Rafael Pass. Cryptography from sublinear-time average-case hardness of time-bounded Kolmogorov complexity. In *Proceedings of the 53rd ACM Symposium on Theory of Computing (STOC)*. ACM, 2021.

[LP21b] Yanyi Liu and Rafael Pass. On one-way functions from NP-complete problems. Manuscript, 2021.

[Nan21] Mikito Nanashima. On basing auxiliary-input cryptography on NP-hardness via nonadaptive black-box reductions. In *12th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 185 of *LIPIcs*, pages 29:1–29:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[OS17] Igor Carboni Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In Ryan O'Donnell, editor, *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, volume 79 of *LIPIcs*, pages 18:1–18:49. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[PF79] Nicholas Pippenger and Michael J. Fischer. Relations among complexity measures. *J. ACM*, 26(2):361–381, 1979.

[Pic20] Ján Pich. Learning algorithms from circuit lower bounds. *CoRR*, abs/2012.14095, 2020.

[San20] Rahul Santhanam. Pseudorandomness and the Minimum Circuit Size Problem. In *11th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 151 of *LIPIcs*, pages 68:1–68:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[Val84] Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91. IEEE Computer Society, 1982.

A Hard-on-average problems in NP

We first introduce some useful notation. For a language $L \subseteq \{0,1\}^*$ we define its *characteristic function*, namely $f_L : \{0,1\}^* \to \{0,1\}$, to be a function given by

$$f_L(x) := \begin{cases} 1, & \text{if } x \in L, \\ 0, & \text{otherwise} \end{cases}$$

for all $x \in \{0,1\}^*$.

For sets $K, L \subseteq \{0,1\}^*$, the *disjoint union of K and L*, denoted $K \uplus L$, is the set $\{0x \mid x \in K\} \cup \{1x \mid x \in L\}$.

For a failure parameter function $\alpha : \mathbb{N} \to [0,1]$, we say that a language L is *α-hard-on-average (α-HoA)* if its characteristic function f_L is α-HoA. Similarly we define average-case easiness for languages.

We prove the following.

Proposition A.1. *Let L be a language in* NP *that is α-HoA for some failure parameter function $\alpha : \mathbb{N} \to [0,1]$. Then, the language $L^* := L \uplus \mathrm{SAT}$ is* NP*-complete and α^*-HoA, where $\alpha^* : \mathbb{N} \to [0,1]$ is a failure parameter function such that $\alpha^*(n) := \alpha(n-1) - 1/2$ for all naturals $n \geq 2$.*

Before we prove Proposition A.1, we recount the following basic observation.

Lemma A.2. NP *is closed under disjoint union.*

We now turn to the proof of Proposition A.1.

Proof of Proposition A.1. By Lemma A.2, the language L^* is in NP since L^* is the disjoint union of $L \in \mathsf{NP}$ and $\mathrm{SAT} \in \mathsf{NP}$.

We will now show that L^* is NP-hard, by giving a polynomial-time reduction R from SAT to L^*. For all $x \in \{0,1\}^*$, let $R(x) := 1x \in \{0,1\}^*$. We see that R is polynomial-time computable. Moreover, if $x \in \mathrm{SAT}$, then $R(x) = 1x \in L^*$, and if $R(x) \in L^*$, then $1x \in L^*$ and so $x \in \mathrm{SAT}$.

What is left is to prove that L^* is α^*-HoA, where $\alpha^* : \mathbb{N} \to [0,1]$ is such that $\alpha^*(n) := \alpha(n-1) - 1/2$ for all naturals $n \geq 2$. Towards a contradiction, assume that L^* is $(1-\alpha^*)$-EoA and let H^* be a heuristic that witnesses this phenomenon. We will give a heuristic H that witnesses the fact that L is $(1-\alpha)$-EoA, whereby establishing the desired contradiction. To this end, let

$$H(x) := H^*(0x)$$

for all $x \in \{0,1\}^*$. We will show that H has the desired average-case performance. That is,

$$\begin{aligned}
\mathop{\mathbf{Pr}}_{x\sim\{0,1\}^n}[H(x) = f_L(x)] &= \mathop{\mathbf{Pr}}_{x\sim\{0,1\}^n}[H^*(0x) = f_{L^*}(0x)] \\
&= \mathop{\mathbf{Pr}}_{y\sim\{0,1\}^{n+1}}[H^*(y) = f_{L^*}(y) \mid y_1 = 0] \\
&\geq \mathop{\mathbf{Pr}}_{y\sim\{0,1\}^{n+1}}[H^*(y) = f_{L^*}(y)] - \mathop{\mathbf{Pr}}_{y\sim\{0,1\}^{n+1}}[y_1 = 1] \\
&\geq 1 - \alpha^*(n+1) - \frac{1}{2} \\
&= 1 - \left(\alpha((n+1)-1) - \frac{1}{2}\right) - \frac{1}{2} \\
&= 1 - \alpha(n)\,.
\end{aligned}$$

□

B McKTP is NP-complete under randomized reductions

In this section, we prove Theorem 1.2 by adapting Ilango's work [Ila20].

B.1 Set Cover

We first fix some notation about Set Cover.

Definition B.1. *The* Set Cover *problem is defined as follows.*

- *Input: A tuple $(n, S_1, \ldots, S_t)$ in binary, where $n \in \mathbb{N}$ and $S_1, \ldots, S_t \subseteq [n]$ are sets such that $[n] \subseteq \bigcup_{i=1}^t S_i$.*
- *Output: The value of*

$$\min_{I\subseteq[t]} \left\{ |I| \mid [n] \subseteq \bigcup_{i\in I} S_i \right\}.$$

Dinur and Steurer [DS14] show that it is NP-hard to approximate Set Cover.

Theorem B.2 ([DS14]). *It is* NP*-hard to approximate Set Cover by a factor of at most $(1-o(1))\ln n$.*

B.2 Approximation algorithms

In the following, we will adopt the following notion of an approximation algorithm.

Definition B.3. *Let Π be an optimization problem. For all instances $I \in \{0,1\}^*$ of Π, let the optimal solution of I be denoted by $\mathrm{OPT}(I) \in \mathbb{R}$. Let $\alpha > 0$. We say that a probabilistic algorithm A* approximates Π by a factor of α *if, for all instances I of Π, it is the case that*

$$\mathrm{OPT}(I) < A(I) \leq \alpha \cdot \mathrm{OPT}(I)$$

with probability at least $1 - o(1)$ over the internal randomness of A.

B.3 Proof of Theorem 1.2

For a string b of length m and a set $R \subseteq [m]$, let $b_{\langle R \rangle}$ be the string of length m where

$$b_{\langle R \rangle}(j) := \begin{cases} b(j)\,, & \text{if } j \in R, \\ 0, & \text{otherwise} \end{cases}$$

for all $1 \leq j \leq m$. Equivalently,

$$b_{\langle R \rangle}(j) := b(j) \wedge \mathbb{1}_{j \in R}$$

for all $j \in [m]$.

Next, we define a uniformly random partition $\mathcal{P} = (P_1, \ldots, P_n)$ of $[m]$ into n parts to be such that each element $i \in [m]$ is put into P_j where $j \in [n]$ is chosen uniformly at random. It will be also useful to think of $\mathcal{P}$ as a uniformly random function $P : [m] \to [n]$.

For a partition $\mathcal{P} = (P_1, \ldots, P_n)$ of $[m]$ and any set $S \subseteq [n]$, we define the *$\mathcal{P}$-lift of S*, denoted $S^{\mathcal{P}}$, to be the set

$$S^{\mathcal{P}} := \bigcup_{i \in S} P_i.$$

Following Ilango [Ila20], we show that McKTP can be used to approximate Set Cover.

Lemma B.4 (Following Ilango [Ila20])**.** *Let $S_1, \ldots, S_t \subseteq [n]$ be sets that cover $[n]$. Let b be a string of length $m \geq (nt)^5$ and let $\mathcal{P} = (P_1, \ldots, P_n)$ be a uniformly random partition of $[m]$ into n parts. Define the oracle $O : \{0,1\}^{\log t} \times \{0,1\}^{\log m} \to \{0,1\}$ to be such that*

$$O(i,z) := \begin{cases} b_{\langle S_i^{\mathcal{P}} \rangle}(z)\,, & \textit{if } i \in [t], \\ 0, & \textit{otherwise,} \end{cases}$$

for all $i \in [t]$ and $z \in [m]$. Let y be the truth table of O, and note that $|y| = mt$. Let ℓ be the size of an optimal cover of $[n]$ by $S_1, \ldots, S_t$. Then, we have that

1. $\mathrm{KT}(b \mid y) \leq 200\ell\,(\log t + \log m)$ *and*
2. $\mathrm{KT}(b \mid y) > \ell\,(\log t + \log m)\,/2$ *with high probability over the choice of b.*

Proof. We prove each item of Lemma B.4 separately.

Claim B.5. *It is the case that $\mathrm{KT}(b \mid y) \leq 200\ell\,(\log t + \log m)$.*

Proof. Assume that an optimal set cover of size ℓ is realized by the sets $S_{i_1}, \ldots, S_{i_\ell}$. Fix some UTM U that has oracle access to y. Let $\Pi \in \{0,1\}^*$ be a program that contains in its description encodings of $i_1, \ldots, i_\ell \in \{0,1\}^t$ and operates as follows:

On input $x \in \{0,1\}^{\log m}$, compute and output $y_{(i_1,x)} \vee \cdots \vee y_{(i_\ell,x)}$.

Note that $|\Pi| \leq (\ell+2)\log t + O(1) \leq 100\ell \log t$. In what follows, let $T \in \mathbb{N}$ be a sufficiently large run-time bound such that

$$\begin{aligned} U^{\Pi,y}\left(x, 1^T\right) &:= y_{(i_1,x)} \vee \cdots \vee y_{(i_\ell,x)} \\ &= O(i_1,x) \vee \cdots \vee O(i_\ell,x) \\ &= \bigvee_{i\in[\ell]} \bigvee_{j \in S_i} b_{\langle P_j \rangle}(x) \\ &= \bigvee_{j\in[n]} b_{P_j}(x) \\ &= b(x)\,, \end{aligned}$$

for all $x \in \{0,1\}^{\log m}$. Note that $T \leq 100\ell\,(\log t + \log m)$. Therefore, we have that $\mathrm{KT}(b \mid y) \leq 200\ell\,(\log t + \log m)$. (Claim B.5) □

We now turn to the lower bound. We do this by a union bound argument. Fix some oracle program $M^y(\cdot) := U^{\Pi,y}\left(\cdot, 1^T\right)$ of program Π that uses oracle y and runs in time T such that $|\Pi| + T \leq \ell\,(\log t + \log m)/2$. Then, as each oracle query requires time $\log t + \log m$, we can deduce that M makes at most $\ell/2 \leq n/2 \leq n$ oracle queries to y.

We will show that

$$\mathop{\mathbf{Pr}}_{b,\mathcal{P}}[M^y \text{ computes } b \text{ in time } T\text{, and } |\Pi| + T \leq \ell\,(\log t + \log m)/2]$$

is exponentially small. We do this by finding a long sequence of inputs $x_1, \ldots, x_d$ on which M has not too large a chance of computing b.

We construct this list recursively, as follows. Let $x_1 := 0^{\log m}$, and let

$$Q_1 := \left\{x \in \{0,1\}^{\log m} \mid M^y(x_1) \text{ makes a query of the form } (i,x) \text{ to } y\text{, for some } i \in [t]\right\}.$$

Now, for $j \geq 1$, if $\{0,1\}^{\log m} \setminus Q_j$ is non-empty, then let x_{j+1} be an element of $\{0,1\}^{\log m} \setminus Q_j$, and let

$$Q_{j+1} := Q_j \cup \left\{x \in \{0,1\}^{\log m} \mid M^y(x_{j+1}) \text{ makes a query of the form } (i,x) \text{ to } y\text{, for some } i \in [t]\right\}.$$

If $\{0,1\}^{\log m} = Q_j$, then terminate the sequence. Since M makes at most n queries to y, we know that $|Q_j| \leq jn$. Thus, since

$$|Q_d| = \left|\{0,1\}^{\log m}\right| = m$$

the length of this sequence is at least m/n. That is, $d \geq m/n$.

It remains to bound the probability

$$\mathbf{Pr}[\text{for all } j \in [d],\ M^y(x_j) = b(x_j)] = \prod_{j=1}^{d} \mathbf{Pr}\left[M^y(x_j) = b(x_j) \mid \bigwedge_{k\in[j-1]} M^y(x_k) = b(x_k)\right].$$

Fix some $j \in [d]$. We will bound

$$\mathbf{Pr}\left[M^y(x_j) = b(x_j) \mid \bigwedge_{k\in[j-1]} M^y(x_k) = b(x_k)\right].$$

Let $E := \bigwedge_{k\in[j-1]} M^y(x_k) = b(x_k)$ be the event that we are conditioning on.

Claim B.6. *It is the case that*

$$\mathbf{Pr}[M^y(x_j) = b(x_j) \mid E] \leq 1 - \frac{1}{2n}.$$

Proof. By construction of the sequence $x_1, \dots, x_d$, we know that on all the inputs $x_1, \dots, x_{j-1}$, the program M^y does not make an oracle call of the form (i, x_j) for any i. Thus, the only time the value of O depends on $b(x_j)$ and $P(x_j)$ is on inputs of the form (i, x_j) for some i, and since $b(x_j)$ and $P(x_j)$ are chosen independently at random, we know that $b(x_j)$ and $P(x_j)$ are still uniform random variables conditioned on E. That is,

$$\mathbf{Pr}[b(x_j) = 1 \mid E] = \frac{1}{2}$$

and

$$\mathbf{Pr}[P(x_j) = r \mid E] = \frac{1}{n}$$

for all $r \in [n]$.

Now, define O' as

$$O'(i,x) := \begin{cases} 0, & \text{if } x = x_j, \\ O(i,x), & \text{otherwise,} \end{cases}$$

and let y' be the truth table of O'. Let also $i_1, \dots, i_v$ with $v \leq \ell/2$ be such that, using the modified oracle O', they are the only oracle queries $M^{y'}(x_j)$ makes that have x_j as the 2nd component of the query, so the queries are $(i_1, x_j), \dots, (i_v, x_j)$. Since $v < \ell$ there exists an element r^* that is not in $S_{i_1} \cup \dots \cup S_{i_v}$.

Moreover, observe that if $P(x_j) = r^*$, then $M^y(x_j)$ will actually make the same oracle queries (and get the same zero responses) as the modified oracle program $M^{y'}$. In this case, since $P(x_j) = r^*$ is not in $S_{i_1} \cup \dots \cup S_{i_v}$, it follows that

$$O(i_1, x_j) = \dots = O(i_v, x_j) = 0$$

regardless of the value of $b(x_j)$. Thus, the output of M^y on input x does not depend at all on the value of $b(x)$ if $P(x_j) = r^*$. Hence, the probability it correctly guesses $M^y(x) = b(x)$ is at most half when $P(x_j) = r^*$.

Since $P(x_j)$ is chosen uniformly at random, we have that $P(x_j) = r^*$ with probability $1/n$. Therefore,

$$\mathbf{Pr}[M^y(x_j) = b(x_j) \mid E] \leq 1 - \frac{1}{2n}. \qquad \text{(Claim B.6)} \ \square$$

Using Claim B.6, we have

$$\begin{aligned} \prod_{j=1}^{d} \mathbf{Pr}\left[M^y(x_j) = b(x_j) \mid \bigwedge_{k \in [j-1]} M^y(x_k) = b(x_k) \right] &\leq \left(1 - \frac{1}{2n}\right)^d \\ &\leq e^{-d/(2n)} \\ &\leq e^{-m/\left(2n^2\right)} \\ &\leq e^{-n^3 t^5/2}. \end{aligned}$$

On the other hand the number of oracle programs of size at most $\ell\left(\log t + \log m\right)/2 \leq O(nt \log n)$ is at most $2^{O\left(n^2 t\right)}$. Thus, by a union bound, the probability that there exists an oracle program Π that computes any bit of b in time T, whereby $|\Pi| + T \leq \ell\left(\log t + \log m\right)/2$, is $o(1)$ as desired. $\square$

Lemma B.4 implies the following corollary.

Corollary B.7. *There is a polynomial-time computable function $M : \mathbb{N} \to \mathbb{N}$ such that the following hold. Given a Set Cover instance $I := (n, S_1, \dots, S_t)$, a random b of length $N \geq (nt)^5$ and a random partition P of $[N]$ into n parts, if one constructs a string y as in Lemma B.4, whereby $|y| \leq M(N)$, then* $\mathrm{KT}(b \mid y)$ *approximates Set Cover by a factor of* 400 *according to Definition B.3. That is, if ℓ is the size of an optimal set cover of I and $c := \log N + \log t$, then it is the case that with probability* 1

$$\frac{2}{c} \cdot \mathrm{KT}(b \mid y) \leq 400\ell,$$

and with probability $1 - o(1)$

$$\frac{2}{c} \cdot \mathrm{KT}(b \mid y) > \ell.$$

Proof. Let $y \in \{0,1\}^*$, $n \in \mathbb{N}$, and $t \in \mathbb{N}$ be as in Lemma B.4. Let $\gamma := 1/2$. Then, $\mathrm{McKT}^M\mathrm{P}$ of dimension $N := |b| \geq (nt)^5$ and $M := N^{1+\gamma} = N^{1+1/2} = N \cdot N^{1/2} \geq Nt = |y|$ is such that Lemma B.4 immediately implies that

$$\ell < \frac{2}{c} \cdot \mathrm{KT}(b \mid y) \leq 400\ell,$$

where the first inequality holds with probability $1 - o(1)$ and the second one holds with probability 1. □

Theorem B.2 and Corollary B.7 yield the following corollary.

Corollary B.8. *There exists a polynomial-time computable function* $m : \mathbb{N} \to \mathbb{N}$ *such that* $\mathrm{McKT}^m\mathrm{P}$ *is* NP*-hard under polynomial-time randomized reductions.*

Finally, by combining Lemma 2.14 and Corollary B.8 we get a proof of Theorem 1.2.

Corollary B.9 (Theorem 1.2, restated)**.** *There exists a polynomial-time computable function* $m : \mathbb{N} \to \mathbb{N}$ *such that* $\mathrm{McKT}^m\mathrm{P}$ *is* NP*-complete under polynomial-time randomized reductions.*

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il