$\mathbf{AC^0}[p]$ Lower Bounds against MCSP via the Coin Problem

Alexander Golovnev* Harvard University
Rahul Ilango† Rutgers University
Russell Impagliazzo‡ University of California, San Diego
Valentine Kabanets§ Simon Fraser University
Antonina Kolokolova¶ Memorial University of Newfoundland
Avishay Tal‖ Stanford University

April 28, 2019

Abstract

Minimum Circuit Size Problem (MCSP) asks to decide if a given truth table of an n-variate boolean function has circuit complexity less than a given parameter s. We prove that MCSP is hard for constant-depth circuits with mod p gates, for any prime $p \geq 2$ (the circuit class $\mathsf{AC}^0[p]$). Namely, we show that MCSP requires d-depth $\mathsf{AC}^0[p]$ circuits of size at least $\exp(N^{0.49/d})$, where $N = 2^n$ is the size of an input truth table of an n-variate boolean function. Our circuit lower bound proof shows that MCSP can solve the coin problem: distinguish uniformly random N-bit strings from those generated using independent samples from a biased random coin which is 1 with probability $1/2 + N^{-0.49}$, and 0 otherwise. Solving the coin problem with such parameters is known to require exponentially large $\mathsf{AC}^0[p]$ circuits. Moreover, this also implies that MAJORITY is computable by a non-uniform AC^0 circuit of polynomial size that also has MCSP-oracle gates. The latter has a few other consequences for the complexity of MCSP, e.g., we get that any boolean function in NC^1 (i.e., computable by a polynomial-size formula) can also be computed by a non-uniform polynomial-size AC^0 circuit with MCSP-oracle gates.

Keywords: Minimum Circuit Size Problem (MCSP), circuit lower bounds, $\mathsf{AC}^0[p]$, coin problem, hybrid argument, MKTP, biased random boolean functions

1 Introduction

Minimum Circuit Size Problem (MCSP) asks to decide if a given boolean function $f\colon \{0,1\}^n \to \{0,1\}$ (presented by its truth table of length $N = 2^n$) can be computed by a boolean circuit of size at most s, for a given parameter $0 \leq s \leq 2^n$. There is no nontrivial algorithm currently known for MCSP other than the "brute force" enumeration of all circuits of size up to s and checking if any one of them computes f. On the other hand, while MCSP is obviously in NP, it is a major open question to decide if MCSP is NP-complete (and there is a growing list of research papers providing arguments for and against various NP-completeness reductions to MCSP [KC00, ABK+06, AHM+08, AD14, AHK15, HP15, HW16, MW17]).

*alexgolovnev@gmail.com. Supported by a Rabin Postdoctoral Fellowship.
†rahul.ilango@rutgers.edu.
‡russell@cs.ucsd.edu. Work supported by a Simons Investigator Award from the Simons Foundation.
§kabanets@cs.sfu.ca. Supported in part by an NSERC Discovery grant.
¶kol@mun.ca. Supported in part by an NSERC Discovery grant.
‖avishay.tal@gmail.com. Supported by a Motwani Postdoctoral Fellowship and by NSF grant CCF-1763299. Part of this work was done while the last four authors were visiting Simons Institute for the Theory of Computing.

Another natural question is to prove circuit lower bounds (for restricted circuit models) for MCSP. Here some results are known. Allender et al. [ABK+06] showed that MCSP requires super-polynomial-size AC^0 circuits (constant-depth circuits with AND, OR, and NOT gates). Hirahara and Santhanam [HS17] proved that MCSP requires almost quadratic-size De Morgan formulas.

It was an open question [AH17, OS17] to prove that MCSP requires super-polynomial $\mathsf{AC}^0[p]$ circuits (constant-depth circuits with AND, OR, NOT and mod p counting gates), for a prime $p > 0$. We resolve this question in the present paper. Our main result is that MCSP requires d-depth $\mathsf{AC}^0[p]$ circuits of size at least $\exp(N^{0.49/d})$, where $N = 2^n$ is the size of an input truth table of an n-variate boolean function.

Previous proof methods of circuit lower bounds for MCSP. The lack of NP-completeness reductions to MCSP and the scarcity of circuit lower bounds for MCSP underscore the general phenomenon that there are very few known reductions to MCSP. The main (if not the only one) use of MCSP inside known reductions is to "break" pseudorandom function generators, an idea going back to the celebrated paper of Razborov and Rudich [RR97] on "natural proofs". The point is that known candidate constructions of pseudorandom function generators produce pseudorandom functions that do have "small" circuit complexity, whereas truly random functions are known to require "high" circuit complexity. Thus, an assumed efficient MCSP algorithm can distinguish between the truth tables of such pseudorandom functions and those of truly random functions, thereby "breaking" the pseudorandom function generator.

Both previously known circuit lower bounds for MCSP by Allender et al. [ABK+06] and by Hirahara and Santhanam [HS17] used MCSP's ability to break pseudorandom function generators together with the existence of known pseudorandom (function) generators that are provably secure against AC^0 and quadratic-size De Morgan formulas, respectively. The same approach cannot be applied to the case of $\mathsf{AC}^0[p]$ circuits as we currently do not have any strong enough pseudorandom generators secure against $\mathsf{AC}^0[p]$!

Our approach. Our approach instead is to reduce the Majority function to MCSP, and use the known $\mathsf{AC}^0[p]$ lower bounds against Majority [Raz87, Smo87]. In fact, we give a reduction to MCSP from the coin problem where one is asked to distinguish between an N-bit uniformly random string, and an N-bit string sampled so that each bit is independently set to 1 with probability $1/2 - \epsilon$ (and to 0 otherwise), for some parameter $\epsilon > 0$. We then use the result of Shaltiel and Viola [SV10] showing that any algorithm solving such a coin problem yields an efficient algorithm for computing the Majority function on inputs of length $1/\epsilon$. To conclude super-polynomial-size $\mathsf{AC}^0[p]$ circuit lower bounds for MCSP from the known lower bounds for the Majority function, we need to be able to solve the coin problem for N-bit strings with the parameter $\epsilon < 1/\mathrm{poly}(\log N)$.

Here is some intuition why MCSP could be useful for solving the coin problem. For $N = 2^n$, an N-bit random string has binary entropy N. On the other hand, N-bit strings sampled using a biased coin with probability $p = 1/2 - \epsilon$ of being 1 would likely have close to $pN < N/2$ ones only, and so come from a smaller set of about $\binom{N}{pN} \approx 2^{H(p)\cdot N}$ strings of size N, where H is the binary entropy function. Information-theoretically, we can describe each string with at most pN ones using about $H(p) \cdot N$ bits. For $p \ll 1/2$, we have that $H(p) \cdot N \ll N$, and so most biased functions have a description of bit complexity much less than N. If somehow we could extend this information-theoretic argument to show that most biased functions will have *circuit complexity* noticeably smaller than that of random functions, we'd be done because MCSP would be able to distinguish between random functions (of higher circuit complexity) and random biased functions (of lower circuit complexity).

Lupanov in 1965 [Lup65] proved that, indeed, biased random functions have circuit complexity smaller than that of random boolean functions. However, Lupanov's result applies only to the case of bias probability $p = 1/2 - \epsilon$ for large (close to constant) ϵ only, and doesn't give anything useful for our case of $\epsilon < 1/\text{poly}(\log N)$. To circumvent the lack of tighter circuit upper bounds for slightly biased random functions, we employ two new ideas.

First, we show that the circuit complexity of q-random functions is very *tightly concentrated* around its expectation, for every probability q. This can be proved using a simple martingale argument (McDiarmid's Inequality [McD89]). The point is that the circuit complexity of a given n-variate boolean function f changes by at most $O(n)$ when we change the value of f on exactly one n-bit input (which can be simply hard-wired into the new circuit for the modified function).

Secondly, we use a *hybrid argument*. By Lupanov's result [Lup65], one can show that for $p = 0.01$, almost all p-biased random functions will have circuit complexity noticeably smaller that $2^n/n$, and in particular, the *expected* circuit complexity of a p-biased random function is at most $0.1 \cdot 2^n/n$. On the other hand, for $p' = 1/2$, well-known counting arguments show that almost all such random functions have circuit complexity very close to $2^n/n$, and in particular, the expected circuit complexity of a random function is at least $0.9 \cdot 2^n/n$. Imagine we have t equally spaced probabilities between $p = 0.01$ and $p' = 1/2$, for some number t to be chosen. Then by the hybrid argument, there will exist two successive probabilities q and $q' \approx q + 1/t$, where the *expected circuit sizes* for q-random and q'-random functions differ by at least $\Omega(2^n/n)/t$.

By the "concentration around the expected circuit size" result mentioned above, we conclude that MCSP is able to distinguish between most q-random boolean functions and most $q' \approx q + 1/t$-random ones. By re-scaling, we conclude that MCSP is able to distinguish between $1/2 - 1/t$-random function and $1/2$-random ones, i.e., that MCSP solves the coin problem for the bias $\epsilon = 1/t$. Finally, our concentration result is strong enough to allow us to choose $t = 2^{\Omega(n)}$, which yields an exponential $\mathsf{AC}^0[p]$ circuit lower bound for MCSP. (To get the quantitatively strongest circuit lower bound for MCSP, we use the recent $\mathsf{AC}^0[p]$ lower bounds for the coin problem by [LSS+18], rather than apply the reduction from the Majority function to the coin problem from [SV10].)

Other results. We are able to generalize our $\mathsf{AC}^0[p]$ circuit lower bounds to several natural variants of MCSP. For a circuit class $\mathcal{C}$, let $\mathcal{C}$-MCSP denote the MCSP problem asking about the $\mathcal{C}$-type circuit complexity of a given truth table. For example, $\mathcal{C}$ can be the class AC^0, where we ask about the gate complexity of a smallest AC^0 circuit computing a given boolean function, or $\mathcal{C}$ can be the class of boolean formulas, where we ask about the formula (leaf) complexity of a smallest formula. We show that for both such cases of $\mathcal{C}$, the problem $\mathcal{C}$-MCSP requires exponential-size $\mathsf{AC}^0[p]$ circuits. This generalization requires us to re-visit Lupanov's general circuit upper bounds for biased random functions. We provide new "hashing-based" arguments for such circuit constructions that apply to the case of formulas as well as constant-depth circuits and formulas.

As a corollary of our reduction of the coin problem to MCSP and some previous results, we obtain that every function in NC^1 can be computed by a non-uniform AC^0 circuit with MCSP oracle gates. We also show that at least one of the following lower bounds must be true: either $\mathsf{NEXP} \not\subset \mathsf{P/poly}$, or $\mathsf{MCSP} \notin \mathsf{ACC}^0$.

Finally, we give a new coin-problem based proof of $\mathsf{AC}^0[p]$ circuit lower bounds for MKTP, a Kolmogorov-complexity variant of MCSP, re-proving the result of [AH17].

The rest of the paper. We give the necessary definitions and facts in Section 2. We prove the aforementioned circuit complexity concentration result for random biased functions in Section 3, and then prove our $\mathsf{AC}^0[p]$ circuit lower bound for MCSP in Section 4. In Section 5, we give

corollaries of our main result. We generalize our lower bounds to the case of $\mathcal{C}$-MCSP in Section 6. The lower bound for MKTP is given in Section 7. Section 8 lists some open questions.

2 Preliminaries

2.1 Complexity basics

For a boolean function $f\colon \{0,1\}^n \to \{0,1\}$, let $size(f)$ denote the size of a smallest circuit (using AND, OR, and NOT gates) that computes f. Let $size(n)$ be the maximum of $size(f)$, over all n-variate boolean functions f. Finally, define $s_n = \mathbf{E}[size(f)]$ to be the average of circuit complexities over all n-variate boolean functions.

Below, all logarithms are base 2, unless stated otherwise.

Building on the work by Shannon [Sha49], Lupanov [Lup58] proved the following lower and upper bounds for $size(n)$.

Theorem 2.1 (Lupanov [Lup58]). *For all sufficiently large $n \in \mathbb{N}$,*

$$size(n) \leq \frac{2^n}{n} + O\left(\frac{2^n \log n}{n^2}\right).$$

Moreover, all but $o(1)$ fraction of uniformly random n-variate boolean functions f require

$$size(f) \geq \frac{2^n}{n} + \Omega\left(\frac{2^n \log n}{n^2}\right).$$

For the case of n-variate functions with a fixed fraction of 1 inputs, Lupanov [Lup65] proved the following generalization of his earlier bounds.

Theorem 2.2 (Lupanov [Lup65]). *Let $f\colon \{0,1\}^n \to \{0,1\}$ be any boolean function that has the value 1 on $k \leq 2^{n-1}$ inputs, where $k \in \Omega(2^n)$. Then, for all sufficiently large $n \in \mathbb{N}$,*

$$size(f) \leq \frac{\log \binom{2^n}{k}}{\log \log \binom{2^n}{k}} + O\left(\frac{2^n \log n}{n^2}\right).$$

Moreover, all but $o(1)$ fraction of random such functions f require

$$size(f) \geq \frac{\log \binom{2^n}{k}}{\log \log \binom{2^n}{k}} + \Omega\left(\frac{2^n \log n}{n^2}\right).$$

2.2 Probability basics

Theorem 2.3 (McDiarmid's Inequality [McD89]). *Let $X_1, \ldots, X_N \in \{0,1\}$ be independent random variables. Let $f\colon \{0,1\}^n \to \mathbb{R}$ be any function such that, for some function $c = c(N)$, for all $1 \leq i \leq N$ and for all $b_1, \ldots, b_N, \widetilde{b}_i \in \{0,1\}$, it holds that*

$$\left| f(b_1, \ldots, b_N) - f(b_1, \ldots, b_{i-1}, \widetilde{b}_i, b_{i+1}, \ldots, b_N) \right| \leq c.$$

Then, for any $\lambda > 0$,

$$\mathbf{Pr}\left[|f(X_1, \ldots, X_N) - \mathbf{E}\left[f(X_1, \ldots, X_N)\right]| \geq \lambda\right] \leq 2 \cdot \exp\left(\frac{2\lambda^2}{Nc^2}\right).$$

Roughly speaking, the inequality states that with high probability the value of f will be of distance at most $O(\sqrt{N} \cdot c)$ around its mean.

2.3 Coin problem

A coin problem is the problem to distinguish between two coins (boolean-valued random variables), where one coin has probability p of being 1, and the other coin probability $q > p$. Usually, $p = 1/2-\epsilon$ and $q = 1/2+\epsilon$, for some $\epsilon > 0$; or, $p = 1/2-\epsilon$ and $q = 1/2$. By a simple "translation argument", it is possible to show that all of these problems are essentially equivalent. For completeness, we state this argument next.

Claim 2.4 (Translation Argument)**.** *Let $0 < p \leq q < 1$, $\varepsilon > 0$. Suppose C is a circuit of size S that solves the $p(1-\varepsilon)$ versus p coin problem on inputs of length N with advantage α. Then, there exists a circuit $\widetilde{C}$ of size at most S that solves the $q(1-\varepsilon)$ versus q coin problem on inputs of length N with advantage at least α.*

Proof. For $p \in (0,1)$, we denote by μ_p the product distribution on $\{0,1\}^N$ where each bit is independently sampled to be 1 with probability p, and 0 otherwise. Let C be a circuit of size S that solves the $p(1-\varepsilon)$ versus p coin problem with advantage

$$\alpha := \mathop{\mathbf{Pr}}_{x\sim\mu_{p(1-\varepsilon)}}[C(x)] - \mathop{\mathbf{Pr}}_{x\sim\mu_p}[C(x)].$$

We construct a distribution over circuits C' that achieves the same advantage on the $q(1-\varepsilon)$ versus q coin problem. The randomized circuit C' is defined as follows: "For each input bit x_i, let $x'_i = x_i$ with probability p/q and $x'_i = 0$ otherwise. Apply C on $(x'_1, \ldots, x'_N)$." Note that, by design, if x is distributed according to a μ_q then x' is distributed according to μ_p, and if x is distributed according to $\mu_{q(1-\varepsilon)}$, then x' is distributed according to $\mu_{p(1-\varepsilon)}$. We get a distribution over circuits C' that solves the $q(1-\varepsilon)$ versus q coin problem with advantage at least α. By averaging, there must exist a deterministic circuit $\widetilde{C}$ that solves the $q(1-\varepsilon)$ versus q coin problem with advantage at least α. Note that $\widetilde{C}$ is obtained from C' by fixing the internal randomness that C' used to decide for each $1 \leq i \leq N$ whether to set $x'_i = x_i$ or $x'_i = 0$. With those choices fixed, $\widetilde{C}(x_1, \ldots, x_N)$ is just a restriction of $C(x_1, \ldots, x_N)$ where some x_i's are set to 0. Hence, the size of $\widetilde{C}$ is at most that of C, as claimed. □

Theorem 2.5 ([SV10])**.** *Let A be an algorithm that distinguishes, with constant distinguishing probability, between n-bit uniformly random strings, and n-bit strings sampled so that each bit is independently set to 1 with probability $1/2-\epsilon$ (and to 0 otherwise). Then there is a non-uniform AC^0 circuit of size $\mathrm{poly}(n/\epsilon)$ that computes the majority function on binary inputs of length $1/\epsilon$, using A-oracle gates.*

Using the theorem above as well as the well-known lower bound for the majority function against $\mathsf{AC}^0[p]$ circuits, for any constant prime p, we can deduce that any algorithm solving the coin problem with bias ϵ on n-bit inputs requires $\mathsf{AC}^0[p]$ depth d circuits of size at least $\exp((1/\epsilon)^{1/O(d)})$. This lower bound has been recently sharpened.

Theorem 2.6 ([LSS+18])**.** *Let A be a boolean function that distinguishes, with constant distinguishing probability, between n-bit uniformly random strings, and n-bit strings sampled so that each bit is independently set to 1 with probability $1/2-\epsilon$ (and to 0 otherwise). Then any depth d $\mathsf{AC}^0[p]$ circuit computing A must have size at least $\exp(\Omega((1/\epsilon)^{1/(d-1)}))$.*

3 Concentration of Circuit Complexity

For every $n \geq 1$, let μ be any product distribution over $\{0,1\}^N$, where $N = 2^n$. Recall that $size(f)$ is the size of a smallest circuit computing a boolean function f. For each integer $n \geq 1$, define

$$s_n^\mu = \mathbf{E}_{f \sim \mu} [size(f)],$$

the expectation of $size(f)$ over n-variate boolean functions f whose N-bit truth tables are sampled according to μ. We show that a random n-variate boolean function sampled from μ is likely to have its circuit complexity very close to the expected circuit complexity s_n^μ.

Theorem 3.1. *For μ and s_n^μ as defined above, if a boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is chosen at random according to distribution μ, then, with probability at least $1 - 2^{-n}$,*

$$|size(f) - s_n^\mu| \leq \sqrt{N} \cdot n^2.$$

Proof. Observe that for any two n-variate boolean functions f and g that differ on exactly one input $a \in \{0,1\}^n$, we have that $|size(f) - size(g)| \leq O(n)$. Indeed, suppose, without loss of generality, that $size(f) \leq size(g)$. Then we can compute g as follows: "Given $z \in \{0,1\}^n$, check if $z = a$. If so, then output the bit $b = g(a)$. Otherwise, use the circuit for f to output $f(z)$." More precisely, if C is a circuit computing f, then the circuit D for g can be described as

$$D(z) := (b \wedge (\wedge_{i=1}^n (z_i = a_i))) \vee (C(z) \wedge (\vee_{i=1}^n (z_i \neq a_i))), \tag{1}$$

where $(z_i = a_i)$ is defined to be z_i for $a_i = 1$, and the negation $\overline{z_i}$ for $a_i = 0$; and $(z_i \neq a_i)$ is the negation of $(z_i = a_i)$. Clearly, the size of D is that of C plus $O(n)$.

Let $X_1, \ldots, X_N \in \{0,1\}$ be independent random variables sampled from the product distribution μ. We apply McDiarmid's Inequality of Theorem 2.3 to the function $size\colon \{0,1\}^N \to \mathbb{N}$, which, as we just argued, differs by at most $c = O(n)$ on any two truth tables that agree in all but one coordinate. We get the desired concentration result by choosing $\lambda = \sqrt{N} \cdot n^2$. That is, all but $\exp(-n)$ fraction of μ-random n-variate boolean functions f have their circuit size $size(f)$ within $\sqrt{N} \cdot n^2$ of the expected circuit size s_n^μ. □

4 Main theorem

Theorem 4.1. *Let $p \geq 2$ be any prime. For any depth $d > 0$ and large enough input size $N = 2^n$,* MCSP *on N-bit truth tables requires depth d $\mathsf{AC}^0[p]$ circuits of size $\exp(\Omega(N^{0.49/(d-1)}))$.*

Proof. Let $t = \lceil 2^{0.49n} \rceil$. Consider an arithmetic sequence of probabilities $(p_0, p_1, p_2, ..., p_t)$ with $p_0 = 0.01$, $p_t = 0.5$ and $p_i = p_0 + i \cdot 0.49/t$. For each $i = 0, \ldots, t$, let μ^i be the product distribution on $\{0,1\}^N$, where each bit is independently sampled to be 1 with probability p_i, and 0 with probability $1 - p_i$. Let

$$s^i = s_n^{\mu^i} = \mathbf{E}_{f \sim \mu^i} [size(f)].$$

By Lupanov's estimates of Theorem 2.1, we have

$$s^t \geq (1 - o(1)) \cdot \frac{2^n}{n} \geq 0.9 \cdot \frac{2^n}{n}.$$

By the Chernoff bound, almost all n-variate boolean functions sampled according to μ^t will assume the value 1 on at most $k = 0.011 \cdot N$ inputs. By Theorem 2.2, we have

$$s^0 \leq H(0.011) \cdot \frac{2^n}{n} + o\left(\frac{2^n}{n}\right) \leq 0.1 \cdot \frac{2^n}{n},$$

where $H()$ is the binary entropy function. It follows that

$$s^t - s^0 \geq 0.8 \cdot \frac{2^n}{n}.$$

This means that there must be an i such that $s^{i+1} \geq s^i + \Omega(2^{0.51n}/n)$. Let $s^* = (s^i + s^{i+1})/2$. By circuit complexity concentration given in Theorem 3.1, we get that, with high probability, n-variate random boolean functions sampled from μ^i have circuit size smaller than s^*, and those sampled from μ^{i+1} have circuit size larger than s^*. Hence, $\mathsf{MCSP}(x, s^*)$ can distinguish between μ^i and μ^{i+1}, with a constant distinguishing probability.

Finally, assume by contradiction that $\mathsf{MCSP} \in \mathbf{AC^0}[p]$ of size S and depth d. Let n be large enough, and let $N = 2^n$. Then, by fixing the second input of MCSP to s^*, we get an $\mathbf{AC^0}[p]$ circuit that on N-bit inputs solves the coin problem distinguishing between p_i and p_{i+1}. By Claim 2.4, there exists a circuit of size at most S and depth at most d that solves the $(1-\varepsilon)/2$ versus $1/2$ coin problem for $\varepsilon = 1 - p_i/p_{i+1} = \Theta(1/t) = \Theta(N^{-0.49})$. However, the latter implies by Theorem 2.6 that $S \geq \exp(\Omega(\varepsilon^{-1/(d-1)})) = \exp(\Omega(N^{0.49/(d-1)}))$. □

5 Consequences

Below, whenever we talk about circuit classes such as AC^0, ACC^0, and TC^0, we mean *non-uniform* circuit classes.

Using Theorem 2.5, we get the following corollary to Theorem 4.1 regarding the MAJORITY function, denoted MAJ.

Corollary 5.1. $\mathrm{MAJ} \in (\mathsf{AC}^0)^{\mathsf{MCSP}}$.

Combined with the inclusion $\mathsf{NC}^1 \subseteq (\mathsf{TC}^0)^{\mathsf{MCSP}}$ of [OS17], Corollary 5.1 yields the following.

Corollary 5.2. $\mathsf{NC}^1 \subseteq (\mathsf{AC}^0)^{\mathsf{MCSP}}$.

In fact, using the same techniques, we can prove something more general.

Theorem 5.3. *Let $\mathcal{C} \subseteq \mathsf{P/poly}$ be any complexity class that has a complete problem under TC^0-computable reductions that is also random-self-reducible via a TC^0-computable reduction. Then we have*

$$\mathcal{C} \subseteq (\mathsf{AC}^0)^{\mathsf{MCSP}}.$$

Proof sketch. It follows from [CIKK16] that any function $f \in \mathsf{P/poly}$ has a non-uniform $(\mathsf{TC}^0)^{\mathsf{MCSP}}$ circuit C of polynomial size that agrees with f on all but an inverse polynomial fraction of inputs. If f is random-self-reducible via a TC^0 reduction, we can recover from C a new polynomial-size $(\mathsf{TC}^0)^{\mathsf{MCSP}}$ circuit computing f exactly (on all inputs). Applying Corollary 5.1 concludes the proof. □

As the class GapL has Determinant as a complete problem under AC^0 reductions (see [All04] for a survey on logspace counting complexity classes), we get the following.

Corollary 5.4. $\mathsf{GapL} \subseteq (\mathsf{AC}^0)^{\mathsf{MCSP}}$.[1]

[1]The potentially bigger class DET is the class of languages that are NC^1-Turing reducible to computing the determinant of an integer matrix [Coo85]. It is not immediately clear if $\mathsf{DET} \subseteq (\mathsf{AC}^0)^{\mathsf{MCSP}}$. Perhaps, one can use the techniques of [AH17] who showed such a result for MKTP.

The following is a non-uniform version of a similar "Karp-Lipton"-style "collapse" theorem from [IKV18], which we state just for the class EXP.

Theorem 5.5. *If* $\mathsf{EXP} \subseteq \mathsf{P/poly}$, *then* $\mathsf{EXP} \subseteq (\mathsf{AC}^0)^{\mathsf{MCSP}}$.

Proof. Using TC^0-computable locally list-decodable binary codes of [GGH+07], we get that EXP contains a complete language that is random-self-reducible via a TC^0-computable reduction. We then appeal to Theorem 5.3. □

We do not know if MCSP is NP-complete. There is a line of research showing that MCSP (or its variants) can't be NP-complete under very restricted kinds of reductions (e.g., "local" reductions of [MW17]). One corollary of Theorem 5.5 is that it will be difficult to rule out non-uniform Turing AC^0 reductions from SAT to MCSP.

Corollary 5.6. *If* $\mathsf{SAT} \notin (\mathsf{AC}^0)^{\mathsf{MCSP}}$, *then* $\mathsf{EXP} \not\subseteq \mathsf{P/poly}$.

Finally, while we don't know how to disprove that $\mathsf{MCSP} \in \mathsf{ACC}^0$, we get that at least some lower bound must be true.

Corollary 5.7. *Either* $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$, *or* $\mathsf{MCSP} \notin \mathsf{ACC}^0$.

Proof. Towards a contradiction, suppose that both (1) $\mathsf{NEXP} \subseteq \mathsf{P/poly}$, and (2) $\mathsf{MCSP} \in \mathsf{ACC}^0$. By the Easy Witness Lemma of [IKW02], (1) implies that $\mathsf{NEXP} = \mathsf{EXP}$. Then by Theorem 5.5, we get that $\mathsf{NEXP} \subseteq (\mathsf{AC}^0)^{\mathsf{MCSP}}$. Combining this with (2) yields that $\mathsf{NEXP} \subseteq \mathsf{ACC}^0$. But the latter contradicts the known lower bound of [Wil14]. □

6 Generalizations

Here we show that, for a number of typical circuit classes $\mathcal{C}$, our lower bound proof (and a reduction from MAJORITY) works also for $\mathcal{C}$-MCSP. In particular, we will show that both AC^0-MCSP and $\mathsf{Formula}$-MCSP require exponential $\mathsf{AC}^0[p]$ circuit lower bounds.

Our lower bound for MCSP used two main ingredients: (1) circuit size concentration for random (biased) boolean functions, and (2) a noticeable difference between most likely circuit sizes for uniformly random and biased boolean functions (where each bit of the truth table is 1 with a small constant probability, say 0.01).

For property (1), we note that the concentration argument only needs the Lipschitz property of a given circuit size measure, which comes from the fact that changing a boolean function on a single n-bit input may change the circuit size of the function by at most $O(n)$ additive term. This holds for virtually every reasonable circuit model, as the proof of Theorem 3.1 shows; there is a potential increase in depth for constant-depth circuits, but this can be avoided for the case where the circuit size is defined to be the total number of gates in the constant-depth circuit (see the proof of Corollary 6.2).

For property (2), we need a Shannon-style counting argument to show that most random n-variate functions have at least certain size S in a given circuit model $\mathcal{C}$, as well as a Lupanov-style argument that (most) boolean functions with very few (a small constant fraction α of) 1s have $\mathcal{C}$-circuit complexity at most some constant fraction δ of S, for some $0 < \delta < 1$ (dependent on α).

Property (2) is known for the case of boolean circuits, as implied by Lupanov's Theorem 2.2, and is known for formulas, by the work of Pippenger [Pip76]. Moreover, it is possible to use the celebrated constructions of Lupanov, giving tight upper bounds for circuit complexity [Lup58] and formula complexity [Lup62] for all boolean functions, to show that biased random functions have

relatively small circuits as well as small formulas (see the appendix). However, we give a different argument below (based on some hashing ideas) that will allow us to reduce the problem of showing small circuit complexity of a random biased boolean function to the known worst-case upper bounds for boolean function on fewer variables. Using such known worst-case upper bounds for the classes of circuits, formulas, and constant-depth circuits (counting the number of gates), we then obtain the required upper bounds for the circuit complexity of constant-biased random functions in these circuit models.

6.1 Hashing construction

As a warm-up, we provide a hashing-based argument where we use pairwise independent permutations. In the following subsection, we will show how to get rid of actual hash functions, and use appropriate identity functions instead.

First, we explain a single step of our overall construction, and then show how to combine a few such steps to get a relatively small circuit for any given biased function.

Let $\mathcal{H}$ be a family of pairwise-independent permutations from $\{0,1\}^n$ to $\{0,1\}^n$. For $h \in \mathcal{H}$ and any $1 \leq m \leq n$, denote by $h^{[m]}$ the function from $\{0,1\}^n$ to $\{0,1\}^m$ which on any input $x \in \{0,1\}^n$ outputs the first m bits of the output $h(x)$. It is easy to see that for any $m > 0$, the family of functions $h^{[m]}$ is a family of pairwise-independent hash functions. Additionally, for $h \in \mathcal{H}$ and any integer $1 \leq i \leq n$, denote by h^i the function from $\{0,1\}^n$ to $\{0,1\}$ which on any input $x \in \{0,1\}^n$ outputs the ith bit of $h(x)$.

Construction Let $f\colon \{0,1\}^n \to \{0,1\}$ be any boolean function with at most $\alpha \ll 1/2$ fraction of 1s, for some constant $\alpha > 0$. Denote by $A \subseteq \{0,1\}^n$ the set of all inputs $x \in \{0,1\}^n$ such that $f(x) = 1$.

- Pick a random permutation h_1 from $\mathcal{H}$. Set $m = n - \log(1/(2\alpha))$, so that the range of $h_1^{[m]}$ is of size $(2\alpha)2^n$.
- For every given input $a \in A$, the probability that there is another $a' \in A$ ($a' \neq a$) such that $h_1^{[m]}(a) = h_1^{[m]}(a')$ (i.e., a collides with some other element of A) is at most $1/2$. Indeed, the required probability is, by the union bound, at most

$$\sum_{a' \in A \setminus \{a\}} \mathbf{Pr}[h_1^{[m]}(a') = h_1^{[m]}(a)] \leq \alpha 2^n (2\alpha 2^n)^{-1} = 1/2,$$

  where we used the pair-wise independence of hash functions $h^{[m]}$. Call an element $a \in A$ *alone in its bucket* if there is no other $a' \in A$ such that $h_1^{[m]}(a) = h_1^{[m]}(a')$. We get that the expected number of elements in A that are alone in their buckets is at least half of A.
- Next, by averaging, there exists a function $h_1 \in \mathcal{H}$ such that, for at least $1/2$ of the strings $a \in A$, each of these a is alone in its bucket, i.e., no other $a' \in A$ exists such that $h_1^{[m]}(a') = h_1^{[m]}(a)$. Fix such a function h_1.
- Define new boolean functions $g_0, g_1, \ldots, g_{n-m}\colon \{0,1\}^m \to \{0,1\}$ as follows. For $y \in \{0,1\}^m$, define

$$g_0(y) = \begin{cases} 1 & \text{if there exists a unique } a \in A \text{ such that } h_1^{[m]}(a) = y \\ 0 & \text{otherwise.} \end{cases}$$

For each $1 \leq i \leq (n-m)$, define

$$g_i(y) = \begin{cases} h_1^{m+i}(a) & \text{if there exists a unique } a \in A \text{ such that } h_1^{[m]}(a) = y \\ 0 & \text{otherwise.} \end{cases}$$

Note that for any $x \in \{0,1\}^n$, if $g_0(h_1^{[m]}(x)) = 1$ and, for each $i \in [n-m]$, $g_i(h_1^{[m]}(x)) = h_1^{m+i}(x)$, then we know that $x \in A$ and so $f(x) = 1$. Call such an $x \in A$ *decided*. By the above, we know that at least $1/2$ fraction of strings in A are decided, and the correct value of f on these strings is determined by the formula

$$\phi_1(x) = g_0(h^{[m]}(x)) \wedge \bigwedge_{i \in [n-m]} \left(g_i(h^{[m]}(x)) = h^{m+i}(x) \right).$$

This algorithm determines the value of f on at least $1/2$ fraction of strings in A. A natural idea is to continue recursively on the remaining undecided strings in A, where we only have at most $1/2$ of A to work with, and so we set the next value of m to be $n - \log(1/\alpha)$ and re-define A to be the subset of undecided elements in the previous version of A. After at most $t \leq \log(\alpha 2^n) \leq n$ iterations, all strings in A are decided, and so the formula for f is

$$\bigvee_{i=1}^{t} \phi_i(x).$$

The circuit complexity of one step of this algorithm is the circuit size of computing a pairwise independent permutation on $\{0,1\}^n$, plus the circuit sizes of $n-m$ boolean function g on m inputs each. The circuit complexity of an m-variate function g will come from our assumed Lupanov-style circuit upper bound for a given circuit class $\mathcal{C}$. A pairwise independent permutation can be easily computed by a poly(n)-size general circuit or a formula. This circuit size of the permutations will be dominated by the sizes of the functions g_i over all t iterations of the algorithm.

6.2 Hashing construction without hash functions

Here we get rid of pairwise independent permutations (hash functions), but at the expense that our result will apply only to random biased functions (as opposed to the previous section where we could handle every function with not too many 1s in its truth table).

Construction Let $f\colon \{0,1\}^n \to \{0,1\}$ be a random α-biased boolean function, for some constant $0 < \alpha \ll 1/2$.

- Set $\ell = (\log(1/2\alpha))/c$, for some constant $c > 0$ to be determined. Set $m = n - \ell$.

  Partition the set $[n]$ into t sets (for t to be determined) $S_0, S_1, \ldots, S_{t-1}$, where $|S_i| = \ell + i$, for all $0 \leq i \leq t-1$. Think of S_0 to be the last ℓ elements in the string $1, 2, \ldots, n$; S_1 the preceding $\ell + 1$ elements; and so on.

  For a string $x \in \{0,1\}^n$ and a subset $S \subseteq [n]$, we denote by x^S the substring of x restricted to positions in S, and by $x^{\overline{S}}$ the substring restricted to positions in the complement $[n] \setminus S$.

  In iteration i, for $0 \leq i \leq t-1$, a given input $x \in \{0,1\}^n$ will be "hashed" to bucket $x^{\overline{S_i}}$.

- Consider iteration 0. For every fixed input $x \in \{0,1\}^n$, the probability that $f(x) = 1$ and that there is another $x' \in \{0,1\}^n$ in the same bucket $x^{\overline{S_0}}$ such that $f(x') = 1$ is at most

$$p_0 = \alpha \cdot (2^\ell - 1) \cdot \alpha \leq \alpha^2 \cdot 2^\ell,$$

since all x' in the same bucket as x are distinct inputs, and f is an α-biased random function. For x such that $f(x) = 1$, we say that x is *undecided after iteration* 0 if there is another x' in the same bucket as x such that $f(x') = 1$.

- For $0 < i < t$, we say that a string $x \in \{0,1\}^n$ such that $f(x) = 1$ is *undecided after i iterations* if x was undecided after $i-1$ iterations, and there is another string x' in the same bucket (i.e., $x^{\overline{S_i}} = (x')^{\overline{S_i}}$) such that x' was also undecided after $i-1$ iterations (and such that $f(x') = 1$).

- By induction, for every given $x \in \{0,1\}^n$, the probability p_i that x is undecided after i iterations is at most

$$p_i \leq p_{i-1} \cdot (2^{\ell+i} - 1) \cdot p_{i-1} \leq p_{i-1}^2 \cdot 2^{\ell+i}.$$

The reason is, given that x was undecided after $i-1$ iterations, its neighbors x' in bucket $x^{\overline{S_i}}$ are different from all its neighbors in all previous buckets $x^{\overline{S_0}}, \ldots, x^{\overline{S_{i-1}}}$. So the event that some such x' was undecided after $i-1$ iterations is independent from that for x, and its probability is at most p_{i-1}.

Solving the recurrence above, we get that

$$p_t \leq \alpha^{2^{t+1}} \cdot 2^{c2^{t+1} \cdot \ell} \leq (2^{c\ell}\alpha)^{2^{t+1}},$$

for some constant $c > 0$ (independent of ℓ and α). This is the constant c we use in our definition of ℓ, so that

$$2^{c\ell}\alpha \leq 1/2,$$

and hence $p_t \leq 2^{-2^t}$.

- As ℓ is a constant (dependent on α), we can partition the set $[n]$ into at least $t = 2\log n$ sets of size ℓ, $\ell+1$,..., $\ell+t-1$. For this t, we get that $p_t \leq 2^{-n^2}$.

- The expected number of inputs $x \in \{0,1\}^n$ that are still undecided after t iterations is at most $2^{n-n^2} \ll 1$. By Markov's inequality, the probability over the choice of an α-biased function f that there will exist at least some undecided input x after t iterations is at most 2^{n-n^2}. Thus for almost all α-biased functions f, no undecided inputs will remain after t iterations.

Let f be any such function. We will next construct a circuit for f.

- For each iteration $0 \leq i < t$, define new boolean functions $g_0^i, g_1^i, \ldots, g_{\ell+i}^i \colon \{0,1\}^{n-(\ell+i)} \to \{0,1\}$ as follows. For $y \in \{0,1\}^{n-(\ell+i)}$, define

$$g_0^i(y) = \begin{cases} 1 & \text{if } \exists!\ x \in f^{-1}(1) \text{ s.t. } x^{\overline{S_i}} = y, \text{ and } x \text{ was not decided after } (i-1) \text{ iterations} \\ 0 & \text{otherwise.} \end{cases}$$

For each $1 \leq j \leq \ell + i$, define

$$g_j^i(y) = \begin{cases} (x^{S_i})_j & \text{if } \exists!\ x \in f^{-1}(1) \text{ s.t. } x^{\overline{S_i}} = y, \text{ and } x \text{ was not decided after } (i-1) \text{ iterations} \\ 0 & \text{otherwise.} \end{cases}$$

- Define
$$\phi_i(x) = g_0^i(x^{\overline{S_i}}) \wedge \bigwedge_{j \in [\ell+i]} \left(g_j^i(x^{\overline{S_i}}) = (x^{S_i})_j \right).$$

- Finally, we claim that
$$f(x) = \bigvee_{i=0}^{t-1} \phi_i(x).$$

The circuit complexity of one step of this algorithm is the circuit sizes of $\ell+i$ boolean functions g_j on $n-(\ell+i)$ inputs each. The circuit complexity of an m-variate function g will come from our assumed Lupanov-style circuit upper bound for a given circuit class $\mathcal{C}$.

6.3 Small circuit complexity of random biased functions

We use the above construction to prove the following.

Theorem 6.1. *Let $0 < \alpha < 1/2$ be any constant. For all but $o(1)$ fraction of α-biased n-variate random functions f, we have*

1. *circuit-size$(f) \leq O(\alpha \cdot \log 1/\alpha) \cdot \frac{2^n}{n}$,*
2. *(AC^0)-formula-size$(f) \leq O(\alpha \cdot \log 1/\alpha) \cdot \frac{2^n}{\log n}$, and*
3. *AC^0-circuit-size$(f) \leq O(\alpha \cdot \log 1/\alpha) \cdot 2^{n/2}$ (where consider the* gate complexity *of a given constant-depth circuit); moreover, the upper bound is for some fixed depth $d_0 > 0$ (independent of f).*

Proof. We will use the construction $f(x) = \vee_{i=0}^{t} \phi_i(x)$ from the previous subsection. For general circuits, the circuit complexity of ϕ_i is at most
$$(n-m+1-(i-1)) \cdot O\left(\frac{2^{m-(i-1)}}{m-(i-1)} \right) + \text{poly}(n),$$
using the $O(2^n/n)$ circuit size upper bound for all n-variate boolean functions [Lup58]. So, over all t iterations, the circuit size will be at most
$$O(n-m+1) \cdot \sum_{i=1}^{t} \frac{2^{m-(i-1)}}{m-(i-1)} + t \cdot \text{poly}(n).$$
Let t_0 be the smallest integer so that at the end of t_0 iterations $m-(t_0-1) \geq n/2$. We split the sum into two sums as follows:
$$\begin{aligned} \sum_{i=1}^{t} \frac{2^{m-(i-1)}}{m-(i-1)} &= \sum_{i=1}^{t_0} \frac{2^{m-(i-1)}}{m-(i-1)} + \sum_{i=t_0+1}^{t} \frac{2^{m-(i-1)}}{m-(i-1)} \\ &\leq \frac{2}{n} \cdot \sum_{i=1}^{t_0} 2^{m-(i-1)} + \sum_{i=t_0+1}^{t} 2^{m-(i-1)} \\ &\leq \frac{4}{n} \cdot 2^m + O\left(2^{n/2}\right) \\ &\leq (9\alpha) \cdot \frac{2^n}{n}. \end{aligned} \tag{2}$$

Hence, the overall circuit complexity of computing f is at most

$$O(\alpha \cdot \log(1/\alpha)) \cdot \frac{2^n}{n},$$

proving item (1).

To prove item (2), we observe that the construction of the previous subsection actually produces a (fixed constant-depth) formula for a given function f, if we use (constant-depth) formulas for all intermediate functions g_j^i. As every n-variate boolean function has a (fixed constant-depth) formula of size $O(2^n/\log n)$ [Lup62], we can follow similar calculations as in case of item (1) above to conclude the required (constant-depth) formula size upper bound.

Finally, for item (3), we use the fact that every n-variate boolean function has an AC^0 circuit of a fixed depth (at most 3) of size $O(2^{n/2})$, where we count the number of gates in the circuit [Dan96]. We observe that the construction from the previous subsection in fact produces a constant-depth circuit, if we use constant-depth circuits for all intermediate functions g_j^i. Moreover, the depth of the resulting circuit is some fixed constant. So over all t iterations, the circuit size of a given function f will be at most

$$O(n-m+1) \cdot \sum_{i=1}^{t} 2^{(m-(i-1))/2} + t \cdot \mathrm{poly}(n),$$

which simplifies to at most

$$O(\log(1/\alpha)) \cdot O(2^{m/2}) \leq O(\alpha \cdot \log(1/\alpha)) \cdot 2^{n/2},$$

as required. □

Using Theorem 6.1, we conclude the following.

Corollary 6.2. *Let $\mathcal{C}$ be the class of general circuits, or formulas, or constant-depth AC^0 circuits. For any prime $p \geq 2$ and any depth $d > 0$ and large enough input size $N = 2^n$, $\mathcal{C}$-MCSP on N-bit truth tables requires depth d $\mathsf{AC}^0[p]$ circuits of size at least $\exp(\Omega(N^{0.49/(d-1)}))$.*

Proof. As observed earlier, our lower bound proof for MCSP requires three ingredients: the Lipschitz property of the circuit complexity measure, a Shannon-style lower bound on the complexity measure for random n-variate boolean functions, and a $O(\alpha \log(1/\alpha))$ factor smaller upper bound on the complexity measure for random α-biased boolean functions (which can be made an arbitrary constant factor ϵ smaller than the corresponding Shannon-style upper bound by choosing the constant bias $\alpha > 0$ to be sufficiently small).

The Lipschitz property is easily seen to hold for both general circuits and formulas. For constant-depth circuits, where we count the number of gates, it also holds, provided the depth of our circuits is at least 3. We sketch the argument next.

We may assume that the circuit has alternating levels of ANDs and ORs, with negations on the bottom variables level. Without loss of generality, the top gate is an OR. (The other case is symmetric.) Case 1. We want to flip the value on $a \in \{0,1\}^n$ from 0 to 1. Add an AND of $x_i^{a_i}$'s, and feed this AND into the top OR gate. (Use just one extra gate.) Case 2: We want to flip from 1 to 0 on $a \in \{0,1\}^n$. Add an OR of $x_i^{1-a_i}$'s, and feed this OR into every AND-gate just one level below the top OR-gate. (Use just one extra gate.) Note that the depth doesn't change, if the original circuit is of depth $d \geq 3$.

The Shannon-style lower bounds for random n-variate functions are known for general circuits $\Omega(2^n/n)$, formulas $\Omega(2^n/(\log n))$, and constant-depth circuits $\Omega(2^{n/2})$ (see, e.g., [Juk12]). Finally, Theorem 6.1 gives matching upper bounds for biased functions. The proof follows. □

7 Circuit lower bounds for MKTP via the coin problem

Here we show how to re-prove a known $\mathsf{AC}^0[p]$ circuit lower bound for MKTP [AH17], using the coin problem. This was the starting point in our attempt to prove an $\mathsf{AC}^0[p]$ lower bound for MCSP. For MKTP, we managed to show that biased random strings have a noticeably smaller KT complexity than that of uniformly random strings, where the bias $1/2-\epsilon$ can be chosen for a sufficiently small ϵ so that we immediately get an $\mathsf{AC}^0[p]$ circuit lower bound for MKTP using Theorem 2.6. We provide the details next.

We first define the KT complexity [ABK+06]. Fix a universal random-access Turing machine U. The KT complexity of a string $x \in \{0,1\}^N$ is defined as the

$$\min\{|d| + t \mid \forall\, 0 \le i \le N+1 \;\; U^d(i) = x_i \text{ in at most } t \text{ steps}\},$$

and $x_{N+1} = \perp$. In other words, x can be computed at every position i in time at most t by a TM U that has random access to some binary string d, and we want to minimize the total sum of the length of such an auxiliary string d and the time bound t.

MKTP is then naturally defined as follows: Given $x \in \{0,1\}^N$ and a parameter s, decide if the KT complexity of x is at most s.

To prove a super-polynomial $\mathsf{AC}^0[p]$ circuit lower bound for MKTP via the coin problem approach (using Theorem 2.6), it would suffice to show that the MKTP oracle can distinguish between uniformly random N-bit strings and those where each bit is sampled, independently, with probability $1/2-\epsilon$, for some $\epsilon \ll 1/\operatorname{poly}\log N$. We'll show how to do this for $\epsilon = N^{-\gamma}$, for some $\gamma > 0$ (in fact, for $\gamma \approx 1/6$).

What we need is to show that a random biased N-bit string can be compressed to have its KT complexity noticeably smaller than that of a uniformly random N-bit. Let $q = 1/2-\epsilon$ be the probability for sampling each bit to be 1 in a random biased string. By standard concentration bounds, we know that a random biased N-bit string will have, with very high probability, the number of 1s very close to $K = qN$. For the simplicity of exposition, we will assume that our random biased strings have at most $K = qN$ ones in them. We then show that every N-bit string with (at most) K ones has its KT complexity much less than N, which is the lower bound on the KT complexity of a uniformly random N-bit string.

It is natural to think of an N-bit string with K ones as a subset of size K in the universe of size N. As there are exactly $\binom{N}{K}$ such subsets, the minimal bit complexity to represent any one of such subsets is $OPT = \log_2 \binom{N}{K}$. Such an information-theoretically optimal encoding of K-size subsets of the N-size universe is known, and is achieved by using the combinatorial number system where we represent each such subset by the unique number of the form

$$\binom{c_K}{K} + \cdots + \binom{c_2}{2} + \binom{c_1}{1}.$$

In more detail, suppose we have $x \in \{0,1\}^N$ where X has exactly K ones. For the base case, when $K = 0$, we output 0. For $K > 0$, we associate with x an integer number using the following recursive procedure: if $x_N = 1$, then output $\binom{N-1}{K} + R_1$, where R_1 is the recursively computed integer associated with $N-1$-bit prefix of x and the parameter $K-1$; if $x_N = 0$, then output R_0, which is the integer associated with the $N-1$-bit prefix of x and the parameter K.

Note that the final integer associated with a given K-size subset $x \in \{0,1\}^N$ has value at most $\binom{N}{K}$ (using Pascal's identity that $\binom{N}{K} = \binom{N-1}{K-1} + \binom{N-1}{K}$), and so has the optimal bit complexity. The encoding is efficient (as outlined above). The decoding is also efficient: given an integer B encoding some unknown K-size subset $x \in \{0,1\}^N$, if $B \ge \binom{N-1}{K}$, then set $x_N = 1$, and recursively

decode $B - \binom{N-1}{K}$ for a $K-1$-size subset of the $N-1$-size universe; otherwise, set $x_N = 0$, and recursively decode B for a K-size subset of the $N-1$-size universe. The running time of such a decoding algorithm is clearly poly(N), and can be shown to be about $O(N^2)$.

For the KT complexity of a string $x \in \{0,1\}^N$ with K ones, we could define d to be the integer encoding this K-size subset, and then define $U^d(i)$ to be an algorithm that does the decoding of d to get all the bits of x. The problem with this is that the runtime to do a complete decoding of d into x is more than N, which is too much. However, the decoding we do is *global*, recovering all bits x_i simultaneously, whereas we just need to give a *local* decoding algorithm: given i, recover x_i.

Encoding: To get a locally decodable representation for our string $x \in \{0,1\}^N$, we partition x into blocks of size b (for b to be determined). For each block j, $1 \leq j \leq N/b$, let K_i be the number of ones in that block. Note that $\sum_{j=1}^{N/b} K_j = K$. Given K_j, encode each block j information-theoretically optimally as described above, using at most $\log_2 \binom{b}{K_j}$ bits. Write the resulting encodings of all blocks one after the other; add N/b pointers (of at most $\log N$ bits each) that point to the beginnings of the encodings of the blocks; add N/b numbers K_j's to the encoding. We get that the total bit size of the overall encoding of x is at most

$$O(N/b)(\log N + \log b) + \sum_{j=1}^{N/b} \lceil \log_2 \binom{b}{K_j} \rceil.$$

The latter sum is at most

$$\sum_{j=1}^{N/b} \log_2 \binom{b}{K_j} + N/b = \log_2 \prod_{j=1}^{N/b} \binom{b}{K_j} + N/b \leq \log_2 \binom{N}{K} + N/b,$$

since the number of sets with K_j ones in block j is at most the number of all subsets of $[N]$ with K ones. Overall, the encoding size is $OPT + O(N \log N/b)$.

Decoding: Given $i \in [N]$, we first figure out which block j it is in, and then decode that entire block (after looking up its number of ones in K_j and its compressed image). As discussed earlier, the decoding runs in time about $O(b^2)$.

Upper-bounding the KT complexity: To keep the KT complexity of x low, we choose b to be $N^{1/3}$. Then the KT complexity of x is at most $OPT + O(N^{2/3})$.

Finally, we show that $\mathsf{MKTP} \notin \mathsf{AC}^0[p]$ as follows. Recall that we consider random biased strings of length N where the bias probability is $q = 1/2 - \epsilon$. Let $K = qN$ be the expected number of ones in a typical biased string, and let's assume that most biased strings have at most K ones in them (for simplicity). We get that for such a biased string, its KT complexity is at most

$$\log_2 \binom{N}{K} + O(N^{2/3}) \approx H(q) \cdot N + O(N^{2/3}),$$

where H is the binary entropy function. For $q = 1/2 - \epsilon$, we can estimate $H(q) \approx 1 - O(\epsilon^2)$. Thus, to have the KT complexity of a typical q-biased N-bit string to be strictly less than N, we need $N \cdot O(\epsilon^2) - O(N^{2/3}) > 0$, which implies that we need $\epsilon > \Omega(N^{-1/6})$. This implies by Theorem 2.6 that MKTP on inputs of size N requires $\mathsf{AC}^0[p]$ circuits of depth d of size at least $\exp(\Omega(N^{1/6(d-1)}))$.

8 Open questions

We managed to find a work-around the lack of a tighter Lupanov-style upper bound on the circuit complexity of just slightly biased random functions where the bias probability is arbitrarily close to 1/2 (as opposed to the bias probability being a small constant bounded away from 1/2). Our proof would be much more direct and constructive if we had such refinements of Lupanov's circuit upper bounds for biased boolean functions (since then we could have proceeded similarly to our proof for the MKTP case in the previous section). Can one prove such tighter circuit upper bounds?

Our current circuit lower bound applies to MCSP, but doesn't seem to apply for its average-case version, the Razborov-Rudich natural property [RR97]. Can one show such an extension?

Finally, we showed (Corollary 5.7) that either $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$ or $\mathsf{MCSP} \notin \mathsf{ACC}^0$. For the proof, we used the original Easy Witness Lemma of [IKW02], the existence of random-self-reducible problems in EXP, plus the known lower bound that $\mathsf{NEXP} \not\subseteq \mathsf{ACC}^0$ [Wil14]. Given the new Easy Witness Lemma and the improved circuit lower bound that $\mathsf{NQP} = \mathsf{NTIME}[n^{\mathrm{poly}\log n}] \not\subseteq \mathsf{ACC}^0$ [MW18], it is natural to ask the following: Can we show that either $\mathsf{NQP} \not\subseteq \mathsf{P/poly}$ or $\mathsf{MCSP} \notin \mathsf{ACC}^0$? Our current proof techniques rely on the existence of a random-self-reducible problem complete for EXP, and no such problem is known for the class NQP.

Acknowledgements

This work was partly carried out while many of the authors were visiting the Simons Institute for the Theory of Computing in association with the DIMACS/Simons Collaboration on Lower Bounds in Computational Complexity, which is conducted with support from the National Science Foundation. We also thank Chris Umans and Ronen Shaltiel for helpful discussions in the early stages of this project (during the Dagstuhl 2018 workshop on "Algebraic Methods in Complexity"). We thank Eric Allender and Shuichi Hirahara for their comments, and special thanks to Eric for pointing us to the paper of Dančik [Dan96] and the discussion of various circuit and formula complexity measures for constant-depth circuit models.

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.

[AD14] Eric Allender and Bireswar Das. Zero knowledge and circuit minimization. In *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, pages 25–32, 2014.

[AH17] Eric Allender and Shuichi Hirahara. New insights on the (non-)hardness of circuit minimization and related problems. In Kim G. Larsen, Hans L. Bodlaender, and Jean-François Raskin, editors, *42nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2017, August 21-25, 2017 - Aalborg, Denmark*, volume 83 of *LIPIcs*, pages 54:1–54:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[AHK15] Eric Allender, Dhiraj Holden, and Valentine Kabanets. The minimum oracle circuit size problem. In *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, pages 21–33, 2015.

[AHM+08] Eric Allender, Lisa Hellerstein, Paul McCabe, Toniann Pitassi, and Michael E. Saks. Minimizing disjunctive normal form formulas and AC^0 circuits given a truth table. *SIAM J. Comput.*, 38(1):63–84, 2008.

[All04] Eric Allender. Arithmetic circuits and counting complexity classes. In Jan Krajicek, editor, *Complexity of Computations and Proofs, Quaderni di Matematica*, volume 13, pages 33–72. Seconda Universita di Napoli, 2004.

[CIKK16] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 10:1–10:24, 2016.

[Coo85] Stephen A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64(1-3):2–21, 1985.

[Dan96] Vladimir Dančik. Complexity of boolean functions over bases of unbounded fan-in gates. *Information Processing Letters*, 57:31–34, 1996.

[GGH+07] Shafi Goldwasser, Dan Gutfreund, Alexander Healy, Tali Kaufman, and Guy N. Rothblum. Verifying and decoding in constant depth. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 440–449. ACM, 2007.

[HP15] John M. Hitchcock and A. Pavan. On the NP-completeness of the minimum circuit size problem. In *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS*, pages 236–245, 2015.

[HS17] Shuichi Hirahara and Rahul Santhanam. On the average-case complexity of MCSP and its variants. In *CCC*, pages 7:1–7:20, 2017. doi:10.4230/LIPIcs.CCC.2017.7.

[HW16] Shuichi Hirahara and Osamu Watanabe. Limits of minimum circuit size problem as oracle. In *31st Conference on Computational Complexity, CCC*, pages 18:1–18:20, 2016.

[IKV18] Russell Impagliazzo, Valentine Kabanets, and Ilya Volkovich. The power of natural properties as oracles. In Rocco A. Servedio, editor, *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, volume 102 of *LIPIcs*, pages 7:1–7:20. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *J. Comput. Syst. Sci.*, 65(4):672–694, 2002.

[Juk12] Stasys Jukna. *Boolean Function Complexity - Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012.

[KC00] Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *STOC*, pages 73–79, 2000. ECCC:TR99-0045.

[LSS+18] Nutan Limaye, Karteek Sreenivasaiah, Srikanth Srinivasan, Utkarsh Tripathi, and S. Venkitesh. The coin problem in constant depth: Sample complexity and parity gates. *arXiv preprint arXiv:1809.04092*, 2018.

[Lup58] Oleg B. Lupanov. On the synthesis of switching circuits. *Doklady Akademii Nauk SSSR*, 119(1):23–26, 1958. English translation in *Soviet Mathematics Doklady*.

[Lup62] Oleg B. Lupanov. Complexity of formula realization of functions of logical algebra. *Problemy Kibernetiki*, 3:782–811, 1962.

[Lup65] Oleg B. Lupanov. On a certain approach to the synthesis of control systems — the principle of local coding. *Problemy Kibernetiki*, 14:31–110, 1965. (in Russian).

[McD89] Colin McDiarmid. *On the method of bounded differences*, pages 148–188. London Mathematical Society Lecture Note Series. Cambridge University Press, 1989.

[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.

[MW18] Cody Murray and R. Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime: an easy witness lemma for NP and NQP. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 890–901. ACM, 2018.

[OS17] Igor C. Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *Proceedings of the 32nd Computational Complexity Conference*, page 18. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.

[Pip76] Nicholas Pippenger. Information theory and the complexity of boolean functions. *Mathematical systems theory*, 10:129–167, 1976.

[Raz87] Alexander A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Math. Notes*, 41(4):333–338, 1987.

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

[Sha49] Claude E. Shannon. The synthesis of two-terminal switching circuits. *Bell System Tech. J.*, 28:59–98, 1949.

[Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*, pages 77–82, 1987.

[SV10] Ronen Shaltiel and Emanuele Viola. Hardness amplification proofs require majority. *SIAM J. Comput.*, 39(7):3122–3154, 2010.

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

[Wil14] Ryan Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014.

[Yab59a] Sergey V. Yablonski. The algorithmic difficulties of synthesizing minimal switching circuits. *Problemy Kibernetiki*, 2:75–121, 1959. (English translation in Problems of Cybernetics II).

[Yab59b] Sergey V. Yablonski. On the impossibility of eliminating perebor in solving some problems of circuit theory. *Doklady Akademii Nauk SSSR*, 124:44–47, 1959. (English translation in Soviet Mathematics Doklady).

A Biased functions have small circuit complexity

Here we show how to use Lupanov's original construction of optimal circuits and formulas to get the corresponding smaller upper bounds for biased random functions. This provides an alternative proof for circuits and formulas of our Theorem 6.1. (However, to get the result for constant-depth circuits, we seem to need to use some kind of hashing-based argument.)

We will prove the following.

Theorem A.1. *Let $0 < \alpha < 1/2$ be any constant. For all but $o(1)$ fraction of α-biased n-variate random functions f, we have*

$$size(f) \leq H(2\alpha) \cdot \frac{2^n}{n} \cdot (1 + o(1)),$$

where H is the binary entropy function.

First we recall Lupanov's circuit upper bound $\frac{2^n}{n}(1 + o(1))$ for all n-variate boolean functions, and then show a small modification that would yield smaller circuit size for most biased random n-variate functions.

A.1 Original Lupanov's construction

We follow the presentation in Wegener's book [Weg87]. The truth table of a given Boolean function $f(x_1, \ldots, x_n)$ is presented in the so-called (k, s)-*form*: the rows of the truth table are labeled by all binary k-tuples (corresponding to the values of $x_1, \ldots, x_k$), and the columns by all binary $(n - k)$ tuples (corresponding to the values of $x_{k+1}, \ldots, x_n$); thus the value of $f(a_1, \ldots, a_n)$ is at the intersection of the row $(a_1, \ldots, a_k)$ and the column $(a_{k+1}, \ldots, a_n)$. In addition, the rows are divided into p disjoint blocks such that each block contains exactly s rows, except possibly for the last block that may contain $0 < s' \leq s$ rows. For a suitable choice of parameters k and s, we can decompose the given function f into a collection of simpler functions of low circuit complexity so that combining the circuits for these simpler functions yields a circuit for the original function f of size $2^n/n + o(2^n/n)$. Below we give more details.

Let us denote by A_i, $1 \leq i \leq p$, the ith block of s rows (s' rows if $i = p$) in the (k, s)-representation of the truth table of $f(x_1, \ldots, x_n)$. We define the following functions f_i, $1 \leq i \leq p$:

$$f_i(x_1, \ldots, x_n) = \begin{cases} f(x_1, \ldots, x_n) & \text{if } (x_1, \ldots, x_k) \in A_i, \\ 0 & \text{otherwise.} \end{cases}$$

It is clear that $f = f_1 \vee \cdots \vee f_p$.

For $1 \leq i \leq p$ and $w \in \{0,1\}^s$ ($w \in \{0,1\}^{s'}$ for $i = p$), we denote by $B_{i,w}$ the set of those columns in the (k, s)-representation whose intersection with the rows in A_i is equal to w. We define

$$f_{i,w}(x_1, \ldots, x_n) = \begin{cases} f_i(x_1, \ldots, x_n) & \text{if } (x_{k+1}, \ldots, x_n) \in B_{i,w}, \\ 0 & \text{otherwise,} \end{cases}$$

for all $1 \leq i \leq p$ and $w \in \{0,1\}^s$ ($w \in \{0,1\}^{s'}$ for $i = p$). Obviously, f is just a disjunction of all $f_{i,w}$'s.

Further, we split each $f_{i,w}$ into the following two functions:

$$f^1_{i,w}(x_1,\ldots,x_n) = \begin{cases} 1 & \text{if } (x_1,\ldots,x_k) \text{ is the } j\text{th row in } A_i \text{ and } w_j = 1 \text{ (for some } j\text{)}, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$f^2_{i,w}(x_1,\ldots,x_n) = \begin{cases} 1 & \text{if } (x_{k+1},\ldots,x_n) \in B_{i,w}, \\ 0 & \text{otherwise.} \end{cases}$$

We have $f_{i,w} = f^1_{i,w} f^2_{i,w}$.

Finally, we write f as follows:

$$f(x_1,\ldots,x_n) = \vee_i \vee_w [f^1_{i,w}(x_1,\ldots,x_k) f^2_{i,w}(x_{k+1},\ldots,x_n)], \tag{3}$$

where i ranges from 1 to p, and w ranges over all binary strings of length s (s' for $i = p$).

Let us denote by W the upper bound on the number of different binary strings obtained in the intersection of any block A_i with the columns of the (k,s)-representation of f. Computing $f(x_1,\ldots,x_n)$ proceeds in the following steps:

1. Compute all minterms on $\{x_1,\ldots,x_k\}$ and $\{x_{k+1},\ldots,x_n\}$. (Recall that a minterm is a conjunction of literals.) This needs at most $O(2^k + 2^{n-k})$ gates [Weg87, Lemma 4.1 (page 75)].

2. Compute all functions $f^1_{i,w}$ by their DNF's (the minterms are already computed). This needs at most $2^k W$ gates.

3. Compute all functions $f^2_{i,w}$ by their DNF's (the minterms are already computed). This needs at most $p2^{n-k}$ gates.

4. Compute f according to (3) (all $f^1_{i,w}$'s and $f^2_{i,w}$'s are already computed). This needs at most $2pW$ gates.

Thus, recalling that $p \le 2^k/s + 1$, we get that the total number of gates required to compute f is

$$O(2^k + 2^{n-k}) + 2^k W + 2^n/s + 2^{n-k} + W2^{k+1}/s + 2W. \tag{4}$$

Choosing $k = \lceil 3\log n \rceil$ and $s = \lceil n - 5\log n \rceil$ yields a circuit for f of size $\frac{2^n}{n}(1 + O(\frac{\log n}{n}))$; here we use the obvious upper bound $W = 2^s$.

A.2 Circuit complexity of biased functions: Proof of Theorem A.1

Using the same (k,s)-representation as above, Yablonski [Yab59b, Yab59a] observed that it is possible to construct a circuit of size $\sigma\frac{2^n}{n}(1+O(\frac{\log n}{n}))$ for a Boolean function $f(x_1,\ldots,x_n)$, provided that $W \le 2^{\sigma s}$ for some $0 < \sigma \le 1$ that is independent of the value of s. In this case, choosing $k = \lceil 3\log n \rceil$ and $s = \lceil (n - 5\log n)/\sigma \rceil$ will yield a circuit of the required size. This is because all the terms in Eq. (4) but $2^n/s$ will still be $o(2^n/n)$, whereas the dominant term $2^n/s$ will now be about $\sigma \cdot (2^n/n)$.

Proof of Theorem A.1. In our case, we have a random n-variate boolean function f where each bit of its truth table is 1 with probability $0 < \alpha < 1/2$. This means that, for every fixed block A_i of rows in the (k,s)-representation of f, a fixed s-bit column of that block is expected to have αs

ones. By Chernoff's bound, the probability (over random functions f) that this s-bit column has more than $2\alpha s$ ones in it is at most $e^{-(1/3)\alpha s}$. By the union bound, the probability that a fixed column $1 \le j \le 2^{n-k}$ has more that $2\alpha s$ in any of p blocks $A_1, \ldots, A_p$ is at most $p \cdot e^{-(1/3)\alpha s}$. Call such a column j *bad*.

We will have $s \ge n - 5\log n \ge 0.9 \cdot n$, and so the probability that a given column j in the (k,s)-representation is bad is at most $p \cdot 2^{-0.3\alpha n}$, which is at most $2^{-0.29\alpha n}$ for $k = \lceil 3\log n \rceil$.

Hence, the expected number of bad columns in the (k,s)-representation of a random α-biased function f is at most $M = 2^{n-k-0.29\alpha n} \le 2^{n(1-0.29\alpha)}$. By Markov's inequality, the probability that the actual number of bad columns is greater than $n \cdot M$ is at most $1/n$. That is, for all but $o(1)$ fraction of random α-biased n-variate functions f, their (k,s)-representation will have at most $nM \le 2^{n(1-0.28\alpha)}$ bad columns.

For an α-biased function f with few bad columns, define the function f' whose (k,s)-representation is the same as that of f except all the bad columns are replaced by zeros. Clearly, f' is such that, for every block A_i of its (k,s)-representation, each s-bit column in that block has at most $2\alpha s$ ones. It follows that

$$W \le \sum_{i=0}^{2\alpha s} \binom{s}{i} \le 2^{H(2\alpha)\cdot s},$$

where H is the binary entropy function. Let $\sigma = H(2\alpha)$. Then, as noted earlier, the function f' has a circuit of size at most

$$\sigma \cdot \frac{2^n}{n}(1 + o(1)).$$

Finally, to compute f, we use this circuit for f', and encode the inputs of f corresponding to the bad columns where we need to flip the value of f'. The number of such hard-wired inputs is at most

$$2^{n(1-0.28\alpha)} \cdot 2^k \le 2^{n(1-0.27\alpha)}.$$

Overall, we need to hard-wire

$$2^{n(1-0.27\alpha)} \cdot n \le 2^{n(1-0.26\alpha)} \le o(2^n/n)$$

bits, since $\alpha > 0$ is some constant. Thus a circuit for f is of size at most

$$\sigma \cdot \frac{2^n}{n}(1 + o(1)),$$

as required. □

A.3 Formula complexity of biased functions

Here we prove the following.

Theorem A.2. *Let $0 < \alpha < 1/2$ be any constant. For all but $o(1)$ fraction of random α-biased n-variate boolean functions f, the formula size of f is at most*

$$H(2\alpha) \cdot (2 + o(1)) \cdot \frac{2^n}{\log n},$$

where H is the binary entropy function.

Proof. The same (k, s)-representation of a given n-variate boolean function f was used by Lupanov to show that f has formula size at most $(2+o(1)) \cdot (2^n/\log n)$; see, e.g., [Weg87, Theorem 3.2 (page 95)]. The actual formula size is at most

$$2pW + pWks + pW(n-k)\log^2(n-k) + (2+o(1)) \cdot p2^{n-k}(n-k) \cdot (\log^{-1}(n-k)). \tag{5}$$

The desired formula size is obtained by choosing $k = \lceil 2\log n \rceil$ and $s = \lceil n - 3\log n \rceil$.

Note that all terms in Eq. (5) except the last one are $o(2^n/\log n)$. The last, dominant term is

$$(2+o(1)) \cdot \frac{2^n}{\log(n-k)} \cdot \frac{n-k}{s},$$

where we used $p = 2^k/s$. If $W \leq 2^{\sigma s}$, for some constant $\sigma > 0$ (independent of s), then we can set $s = (\lceil n - 3\log n \rceil)/\sigma$, and get the formula size $\sigma \cdot (2+o(1)) \cdot \frac{2^n}{\log n}$.

The rest of the argument is as in the proof of Theorem A.1. □