Electronic Colloquium on Computational Complexity, Report No. 22 (2019)

Circuit Lower Bounds for MCSP from Local Pseudorandom Generators

Mahdi Cheraghchi* Valentine Kabanets† Zhenjian Lu‡ Dimitrios Myrisiotis§

February 23, 2019

Abstract

The Minimum Circuit Size Problem (MCSP) asks if a given truth table of a Boolean function f can be computed by a Boolean circuit of size at most θ, for a given parameter θ. We improve several circuit lower bounds for MCSP, using pseudorandom generators (PRGs) that are local; a PRG is called *local* if its output bit strings, when viewed as the truth table of a Boolean function, can be computed by a Boolean circuit of small size. We get new and improved lower bounds for MCSP that almost match the best-known lower bounds against several circuit models. Specifically, we show that computing MCSP, on functions with a truth table of length N, requires

- $N^{3-o(1)}$-size de Morgan formulas, improving the recent $N^{2-o(1)}$ lower bound by Hirahara and Santhanam (CCC, 2017),
- $N^{2-o(1)}$-size formulas over an arbitrary basis or general branching programs (no non-trivial lower bound was known for MCSP against these models), and
- $2^{\Omega\left(N^{1/(d+2.01)}\right)}$-size depth-$d$ AC^0 circuits, improving the superpolynomial lower bound by Allender et al. (SICOMP, 2006).

The AC^0 lower bound stated above matches the best-known AC^0 lower bound (for PARITY) up to a small *additive* constant in the depth. Also, for the special case of depth-2 circuits (i.e., CNFs or DNFs), we get an almost optimal lower bound of $2^{N^{1-o(1)}}$ for MCSP.

Keywords. Minimum circuit size problem (MCSP), circuit lower bounds, pseudorandom generators (PRGs), local PRGs, de Morgan formulas, branching programs, constant-depth circuits.

*Department of Computing, Imperial College London, London, UK; m.cheraghchi@imperial.ac.uk.
†School of Computing Science, Simon Fraser University, Burnaby, BC, Canada; kabanets@cs.sfu.ca.
‡School of Computing Science, Simon Fraser University, Burnaby, BC, Canada; zhenjian_lu@sfu.ca.
§Department of Computing, Imperial College London, London, UK; d.myrisiotis17@imperial.ac.uk.

ISSN 1433-8092

Contents

1 Introduction

Given the truth table of some Boolean function f and a size parameter θ, the minimum circuit size problem ($\mathtt{MCSP}$) asks whether f can be computed by a circuit of size at most θ. Understanding the exact complexity of $\mathtt{MCSP}$ is an important open problem in computational complexity theory, dating back to the 1950s [Tra84].

It is easy to see that $\mathtt{MCSP}$ is in NP. A popular conjecture is that $\mathtt{MCSP}$ is also NP-hard. However, despite serious efforts over the years, such a proof is still unknown. Given that it is difficult to show that $\mathtt{MCSP}$ is hard, perhaps the problem is easy? It turns out that this cannot be the case under some plausible cryptographic assumptions. More specifically, it is known that if one-way functions exist, then $\mathtt{MCSP}$ is not in P [KC00]. As proving an *unconditional* lower bound for $\mathtt{MCSP}$ seems far beyond the reach of currently known techniques, can we at least prove unconditional lower bounds for $\mathtt{MCSP}$ against some restricted computational models?[1]

Two of the most studied restricted computational models in complexity theory are constant-depth circuits (AC^0) and de Morgan formulas. For AC^0 circuits, the best-known lower bound is about $\mathtt{PARITY}$: $\mathtt{PARITY}$ on N variables requires depth-d AC^0 circuits of size $2^{\Omega\left(N^{1/(d-1)}\right)}$ [Hås86]. For de Morgan formulas, the state-of-the-art lower bound is almost cubic, namely $N^{3-o(1)}$, for some polynomial-time computable function [Hås98, Tal14, Tal17a, DM18].

Notably, there are also lower bounds against these models for $\mathtt{MCSP}$. Allender et al. [ABK+06] showed that $\mathtt{MCSP}$, on functions represented as a truth table of length N, cannot be computed by polynomial-size constant-depth AC^0 circuits. In fact, by a more careful analysis of their argument, one can get a lower bound of $2^{\Omega\left(N^{1/(c\cdot d+O(1))}\right)}$, for a constant $c \geq 2$. However, such a lower bound still has a worse dependence on the depth compared to the $\mathtt{PARITY}$ lower bound. For de Morgan formulas, Hirahara and Santhanam [HS17] showed that computing $\mathtt{MCSP}$ requires de Morgan formulas of size $N^{2-o(1)}$.

Given these two $\mathtt{MCSP}$ lower bounds and the best-known lower bounds against these two models, it is natural to ask whether we can get $\mathtt{MCSP}$ lower bounds against small-depth circuits and de Morgan formulas that match the state-of-the-art lower bounds against these models. More specifically, can we show that computing $\mathtt{MCSP}$ requires depth-d AC^0 circuits of size $2^{\Omega\left(N^{1/(d+O(1))}\right)}$ and de Morgan formulas of size $N^{3-o(1)}$? Furthermore, can we show lower bounds for $\mathtt{MCSP}$ against some other restricted models that match their state-of-the-art lower bounds? In this paper, we answer these questions in the affirmative.

1.1 Our results

Let $\mathtt{MCSP}_N$ denote the minimum circuit size problem on functions with truth table of length N. Our first result is an almost-cubic de Morgan formula lower bound for $\mathtt{MCSP}$.

Theorem 1. *Any de Morgan formula computing* $\mathtt{MCSP}_N$ *has size at least* $N^3/2^{O\left(\log^{2/3} N\right)}$.

We also get almost-quadratic lower bounds against formulas over an arbitrary basis as well as general branching programs; these almost match the best-known lower bounds against these models [Nec66].

[1] A recent line of research on *hardness magnification* [OS18, OPS18] provides another motivation for proving relatively weak lower bounds for restricted circuit models against certain "gap variants" of $\mathtt{MCSP}$. Such lower bounds are shown to imply much stronger (superpolynomial) lower bounds.

Theorem 2. *Let C be either a formula over any basis or a branching program that computes $\mathtt{MCSP}_N$. Then C must have size at least $N^2/2^{O\left(\sqrt{\log N}\right)}$.*

For small-depth circuits, we have the following improved lower bound for MCSP, which its dependence on the depth matches the one in the PARITY lower bound, up to a small additive constant.

Theorem 3. *For every $d > 2$ and every constant $\gamma > 0$, any depth-d AC^0 circuit computing $\mathtt{MCSP}_N$ must have size $2^{\Omega\left(N^{1/(d+2+\gamma)}\right)}$.*

For the special case of depth-2 circuits, we can have an almost optimal lower bound.

Theorem 4. *Any CNF or DNF computing $\mathtt{MCSP}_N$ must have size $2^{N/\widetilde{O}\left(\log^2 N\right)}$.*

Also, in this paper, we give a more fine-grained analysis of the approach of obtaining MCSP lower bounds from average-case hardness via the Nisan-Wigderson framework (Section 7).

1.2 Our techniques

For a class $\mathfrak{C}$ of N-variate Boolean functions, a pseudorandom generator (PRG) against $\mathfrak{C}$ is a deterministic efficiently-computable function G mapping short binary strings (seeds) to longer binary strings so that every function in $\mathfrak{C}$ accepts G's output on a uniformly random seed with about the same probability as that for an actual uniformly random string. A key notion in this work is that of a local PRG. We say that a PRG is *local* if its N-bit output (viewed as the truth table of some function) has small circuit complexity. More precisely, for any fixed seed to the PRG, there exists a small circuit such that, given $j \in [N]$ as an input, the circuit computes the j-th bit of the PRG output, where the complexity of the circuit is measured relative to its input length, namely $\log N$.

Local PRGs in the context of MCSP (and related problems) have been studied in previous works (see, e.g., [ABK+06, OS17, HS17, Hir18]). In this work, we refine the previous approaches, and obtain stronger circuit lower bounds by establishing strong locality properties of certain PRG constructions.

MCSP lower bounds from local PRGs. Suppose we have a local PRG against some class of circuits $\mathfrak{C}$ of size s, and we want to show that MCSP cannot be computed by any size-s circuit in $\mathfrak{C}$. Suppose some size-s circuit C in $\mathfrak{C}$ computes MCSP. Using the fact that a random function has almost maximum circuit complexity, we have that C will output FALSE on most of its inputs (by setting the size parameter θ to be a non-trivial quantity that is asymptotically smaller than $2^n/n$, where n is the input length of the function). If we replace the uniformly random inputs with the outputs of the local PRG, then, by the definition of PRG, C will still output FALSE with large probability. However, since the PRG is local, all of its outputs have circuit complexity smaller than the size parameter θ, and hence must be accepted by C. A contradiction.

To get a strong lower bound, we would like to make the above argument to work for large s. Note that the local complexity of the PRG, $\lambda(s)$, is a function on the size of the circuit C, and we need this local complexity to be "non-trivial" in order to reach a contradiction. Therefore, we want to choose s so that this local complexity remains asymptotically smaller than $2^n/n$. As a result, the final lower bound (i.e., the largest s that we can choose) is determined by the local complexity λ. So the main question we study in our paper is: What is the smallest local complexity of a PRG against a given circuit class?

MCSP lower bound against de Morgan formulas. Our formula lower bound for MCSP is obtained by applying the framework described above to a local PRG against formulas. The state-of-the-art PRG against formulas is given by Impagliazzo, Meka, and Zuckerman [IMZ12], which we refer to as the IMZ PRG. Their PRG has a seed length of $s^{1/3+o(1)}$ for size s formulas (note that such a PRG is useful against sub-cubic formulas only). If we want to utilize the IMZ PRG to get an MCSP lower bound against formulas, we will need to argue that the IMZ PRG is local.

In fact, in order to get an almost-cubic lower bound, we will need such a PRG to be strongly local in the sense that any single output bit of the PRG (on any given fixed seed) can be computed by a circuit of size comparable to its seed length, which is $s^{1/3+o(1)}$. However, by inspecting the construction, the IMZ PRG does not seem to have such a property, and a straightforward implementation seems to require a circuit of size at least $s^{2/3}$ (see Appendix B for more details), which yields a weaker lower bound for MCSP.

To overcome this issue, we present an alternative PRG useful against sub-cubic formulas which is strongly local. The construction of this PRG can be viewed as a modification of the IMZ PRG. At a high level, it is based on the Ajtai-Wigderson construction [AW89], which is a framework for constructing PRGs against computations that can be simplified under (pseudo)random restrictions. This framework is then combined with the ideas for reducing (recycling) random bits using an extractor, by exploiting communication bottlenecks in computations [NZ96]. Our modification, particularly the utilization of the Ajtai-Wigderson construction, allows us to compute any output bit of the PRG efficiently by reducing the number of calls to the extractor. Using some crucial observations on the circuit complexity of certain pseudorandom objects, we get a PRG that is locally computable by a $s^{1/3+o(1)}$-size circuit.[2]

MCSP lower bounds against AC^0. We use a local PRG against AC^0 to get MCSP lower bounds. To get a lower bound matching the one in Theorem 3, we can use the state-of-the-art PRG against AC^0 by Trevisan and Xue [TX13], which has a seed length of $(\log s)^{d+O(1)}$ for size-s depth-d AC^0 circuits. By a careful analysis of the construction of this PRG, we can show that the Trevisan-Xue PRG is strongly local and can be used to get an MCSP lower bound that is close to the one stated in Theorem 3. However, in this paper, we will present a more direct proof of such a lower bound by using the pseudorandom switching lemma for constant-depth circuits, which is due to Trevisan and Xue [TX13] as well, and is a key ingredient in their PRG.

The idea is to show that for any small-depth circuit of size less than the claimed lower bound, there is some locally computable restriction that turns the circuit into a constant function, but leaves many variables unrestricted. However, MCSP cannot be constant under such a restriction, because depending on the partial assignment to the unrestricted variables, the resulting input function (which is composed of the restriction and the partial assignment) can be either easy or hard. Such an approach based on pseudorandom restrictions can also be applied to the special case of depth-2 circuits to get almost optimal CNF (and DNF) lower bounds for MCSP.

Remainder of the paper. We give the necessary background in Section 2. In Section 3, we describe our framework of using local PRGs to obtain lower bounds for MCSP. We prove the almost-cubic de Morgan formula lower bound for MCSP (Theorem 1) in Section 4, and the almost-quadratic

[2] It is also possible to use the original IMZ PRG to obtain an almost-cubic formula lower bound for MCSP. We can show that the IMZ PRG, although not fully strongly local, is "almost strongly local" in the sense that *most* of its outputs have very small circuit complexity; see Appendix B.

lower bounds against formulas over an arbitrary basis and branching programs (Theorem 2) in Section 5. The improved AC^0 lower bounds for $\texttt{MCSP}$ (Theorem 3 and Theorem 4) are proved in Section 6. In Section 7, we discuss the framework of proving $\texttt{MCSP}$ lower bounds from average-case hardness. Finally, we give some open problems in Section 8.

2 Preliminaries

2.1 Notation

For any computational model, we use the term *size* to refer to its complexity measure. For example, if the model is circuits of some fixed depth, then the size is the number of gates in the circuit.

For a positive integer n that is a power of two,[3] we use the following notation:

- $[n]$ denotes the set $\{1, 2, \dots, n\}$. We will sometimes identify $[n]$ with $\{0,1\}^{\log n}$, in the natural way.
- $\mathbb{F}_n$ denotes the field with n elements. Again, we will sometimes identify $\mathbb{F}_n$ with $\{0,1\}^{\log n}$ where the elements in $\mathbb{F}_n$ are represented by $(\log n)$-bit strings.
- U_n denotes the uniform distribution over $\{0,1\}^n$.
- We use $\widetilde{O}(\cdot)$ to hide polylogarithmic factors. That is, for any $f \colon \mathbb{N} \to \mathbb{N}$, we have that $\widetilde{O}(f(n)) = f(n) \cdot \mathsf{polylog}(f(n))$.
- For a function $f \colon \{0,1\}^n \to \{0,1\}$, $\texttt{tt}(f) \in \{0,1\}^{N=2^n}$ denotes the truth table of f, and $\texttt{CC}(f)$ denotes its circuit complexity, that is, the size of the smallest circuit that computes f on all inputs $x \in \{0,1\}^n$.

2.2 Pseudorandomness

Definition 5 (Pseudorandom generators)**.** *Let $G \colon \{0,1\}^r \to \{0,1\}^n$ be a function, $\mathcal{F}$ be a class of Boolean functions, and $0 < \varepsilon < 1$. We say that G is a pseudorandom generator of seed length r that ε-fools $\mathcal{F}$ if, for every function $f \in \mathcal{F}$, it is the case that*

$$\left|\mathbb{E}_{z\sim\{0,1\}^r}[f(G(z))] - \mathbb{E}_{x\sim\{0,1\}^n}[f(x)]\right| \leq \varepsilon.$$

A multidimensional distribution is called *k-wise independent* if any k coordinates of the distribution are uniformly distributed.

Definition 6 (k-wise independence)**.** *A distribution X over $[m]^n$ is called* k-wise independent *if for any $1 \leq i_1 \leq i_2 \leq \cdots \leq i_k \leq n$ and every $b_1, b_2, \dots, b_k \in [m]$, we have*

$$\mathbf{Pr}[X_{i_1} = b_1, X_{i_2} = b_2, \dots, X_{i_k} = b_k] = m^{-k}.$$

[3] We may sometimes implicitly assume that some quantity, such as the number of variables, or circuit size, is a "nice" number (e.g., a power of two). This can always be made true by adding dummy variables or dummy gates, which change the respective quantity by a small amount, and all of our results will still hold asymptotically.

We say that a function G is a *k-independent generator* if, for random inputs, the distribution of the outputs of G is k-wise independent.

We will need efficient and local constructions for k-independent generators as well as some other pseudorandom objects. These objects can be constructed using finite fields. We need the following result, which says that finite field arithmetic can be performed by almost-linear-size circuits.

Fact 7 (See, e.g., [vzGG13, GS13]). *For any integer $\ell > 0$, let the elements in $\mathbb{F}_{2^\ell}$ be represented by ℓ-bit strings. Then, addition over $\mathbb{F}_{2^\ell}$ can be performed by a circuit of size $O(\ell)$ and multiplication over $\mathbb{F}_{2^\ell}$ can be performed by a circuit of size $\widetilde{O}(\ell)$.*

We now describe an efficient construction for k-independent generator, using the fact that finite field arithmetic can be done using *almost linear-size* circuits.

Lemma 8. *For any integer $k > 0$, there exists a k-independent generator $G\colon \{0,1\}^r \to [m]^n$, with $r = k \cdot \max\{\log n, \log m\}$, such that the following holds. There exists a circuit of size*

$$k \cdot \max\{\widetilde{O}(\log n), \widetilde{O}(\log m)\}$$

such that, given $j \in \{0,1\}^{\log n}$ and a seed $z \in \{0,1\}^r$, the circuit computes the j-th coordinate of $G(z)$.

Proof. Let $n' = \max\{n, m\}$ and suppose $n' = 2^\ell$. We view the elements in $\mathbb{F}_{n'}$ as ℓ-bit strings. Consider the following function $g\colon \mathbb{F}_{n'} \times \mathbb{F}_{n'}^k \to \mathbb{F}_{n'}$:

$$g(i, z_0, \ldots, z_{k-1}) = z_0 + z_1 \cdot i + \cdots + z_{k-1} \cdot i^{k-1}.$$

It is known (see Proposition 3.33 of [Vad12]) that the function $G\colon \mathbb{F}_{n'}^k \to \mathbb{F}_{n'}^{n'}$ given as

$$G(z_0, \ldots, z_{k-1}) = \left(g(1, z_0, \ldots, z_{k-1}), g(2, z_0, \ldots, z_{k-1}), \ldots, g(n', z_0, \ldots, z_{k-1})\right),$$

is a k-independent generator.

Using Fact 7 it is easy to implement a circuit of size $k \cdot \widetilde{O}(\ell)$ that computes $g(j, z)$. Note that to get an output in $[m]$ we can simply output the first $\log m$ bits of $G(z)_j$, since the field has characteristic 2. $\square$

We will need a tail bound for k-wise independent distributions.

Proposition 9 (See Problem 3.8 of [Vad12]). *Let $X_1, X_2, \ldots, X_n$ be k-wise independent variables over $\{0,1\}$, and let $\mu := \frac{1}{n} \cdot \mathbb{E}\left[\sum_{i=1}^n X_i\right]$. Then, it is the case that*

$$\mathbf{Pr}\left[\frac{1}{n} \cdot \sum_{i=1}^{n} X_i \notin \left(1 \pm \frac{1}{2}\right) \cdot \mu\right] \le \left(\frac{k^2}{n \cdot \mu^2}\right)^{\lfloor k/2 \rfloor}.$$

The following simple fact will be convenient for us.

Lemma 10. *Let X and Y be two random variables that take values in $\{0,1\}$ and $\mathcal{E}$ be some event. If*

- $|\mathbb{E}[X \mid \mathcal{E}] - \mathbb{E}[Y \mid \mathcal{E}]| \le \varepsilon_1$ *and*
- $\mathbf{Pr}[\neg\mathcal{E}] \le \varepsilon_2$,

then $|\mathbb{E}[X] - \mathbb{E}[Y]| \leq \varepsilon_1 + \varepsilon_2$.

Proof. We have

$$\mathbb{E}[X] = \mathbb{E}[X \mid \mathcal{E}] \cdot \mathbf{Pr}[\mathcal{E}] + \mathbb{E}[X \mid \neg\mathcal{E}] \cdot \mathbf{Pr}[\neg\mathcal{E}],$$

and

$$\mathbb{E}[Y] = \mathbb{E}[Y \mid \mathcal{E}] \cdot \mathbf{Pr}[\mathcal{E}] + \mathbb{E}[Y \mid \neg\mathcal{E}] \cdot \mathbf{Pr}[\neg\mathcal{E}].$$

Then,

$$\begin{aligned}
\mathbb{E}[X] - \mathbb{E}[Y] &= (\mathbb{E}[X \mid \mathcal{E}] - \mathbb{E}[Y \mid \mathcal{E}]) \cdot \mathbf{Pr}[\mathcal{E}] + (\mathbb{E}[X \mid \neg\mathcal{E}] - \mathbb{E}[Y \mid \neg\mathcal{E}]) \cdot \mathbf{Pr}[\neg\mathcal{E}] \\
&\leq |\mathbb{E}[X \mid \mathcal{E}] - \mathbb{E}[Y \mid \mathcal{E}]| + \mathbf{Pr}[\neg\mathcal{E}] \\
&\leq \varepsilon_1 + \varepsilon_2.
\end{aligned}$$

The fact

$$\mathbb{E}[Y] - \mathbb{E}[X] \leq \varepsilon_1 + \varepsilon_2$$

can be similarly shown. □

2.3 Random restrictions

A restriction for a n-variate Boolean function f, usually denoted as $\rho \in \{0, 1, *\}^n$, specifies a way of fixing the values of some subset of variables for f. That is, if ρ_i is $*$, we leave the i-th variable unrestricted and otherwise fix its value to be $\rho_i \in \{0, 1\}$. We denote by f_ρ the restricted function after the variables are restricted according to ρ, and denote by $\rho^{-1}(*)$ the set of unrestricted variables. A random restriction is then a distribution over restrictions. We will often view sampling a random restriction as a two-step process: The first step is selecting (in some random manner) a subset of unrestricted variables (also called the "star" or "$*$" variables) and the second step is fixing (in some random manner) the values of all the other variables. Then, a random restriction over n variables can also be specified by a pair $(\sigma, \beta) \in \{0, 1\}^n \times \{0, 1\}^n$, where σ (as a characteristic string) specifies the set of unrestricted variables, and β specifies the values for fixing the restricted variables.

We say that a random restriction (or random selection) is p-regular if each variable is left unrestricted with probability p. One way to generate a p-regular random restriction is to leave each variable, independently, unrestricted with probability p, and otherwise assign to it a 0 or a 1 uniformly at random. Such a random restriction is called a *(truly) p-random restriction*. Note that to sample such a restriction, we can first pick a string in $\{0, 1\}^{n \cdot \log(1/p)}$ to specify the selection of the unrestricted variables, where a coordinate is unrestricted if and only if all of its corresponding $\log(1/p)$ bits are 0, and then a string in $\{0, 1\}^n$ to specify the values assigned to each of the restricted variables. So sampling a restriction in this way requires $n \cdot \log(1/p) + n$ random bits. We can also generate a restriction in a pseudorandom manner, which may use fewer random bits. For example, one way to do this is to use a limited-independence distribution, so that each variable is set to be unrestricted with probability p, and any k of the variables are independent. Note that such a "pseudorandom selection" can be obtained using a k-wise independent distribution on $[1/p]^n$. Also, we can let each variable be assigned a 0 or a 1 uniformly at random in a way such that any k of the variables are independent; this again can be done using a k-wise independent distribution on $\{0,1\}^n$.

Finally, note that we can also get a restriction by combining a sequence of restrictions $\rho_1, \ldots, \rho_t$, in a natural way, namely by applying the sub-restrictions one by one. In this case, we write the finial restriction as $\rho_1 \circ \cdots \circ \rho_t$.

2.4 Simple facts about Boolean circuits

We refer to a textbook as [Juk12] for a general introduction to Boolean circuits.

Proposition 11. *A Boolean circuit of size s can be specified using $O(s \log s)$ bits. Hence there are at most $2^{O(s\log s)} = s^{O(s)}$ distinct circuits of size at most s.*

Theorem 12 ([Sha49])**.** *The fraction of functions on n variables that have a circuit of size less than $2^n/(3n)$ is $o(1)$.*

Lemma 13. *For any integer $t > 0$, there exists a circuit C of size $\widetilde{O}(t)$ such that, given any string $x \in \{0,1\}^t$, the circuit does the following:*

- *If $x = 0^t$, then C outputs $(0, 0^{\log t})$.*
- *If $x \neq 0^t$, then C outputs $(1, q)$, where $q \in \{0,1\}^{\log t}$ is the index of the first bit in x that is not 0.*

Proof. Define $z^{(0)} = \left(0, 0^{\log t}\right)$ and $z^{(i)}$, for any $i = 1, 2, \ldots, t$, recursively as follows:

$$z^{(i)} = \begin{cases} z^{(i-1)}, & \text{if } (z^{(i-1)})_1 = 1, \\ z^{(i-1)}, & \text{if } (z^{(i-1)})_1 = 0 \text{ and } x_i = 0, \text{ and} \\ (1, i), & \text{if } (z^{(i-1)})_1 = 0 \text{ and } x_i = 1. \end{cases}$$

Note that each $z^{(i)}$ can be computed in $\mathsf{polylog}(t)$ size given $z^{(i-1)}$. Using a circuit of size $\widetilde{O}(t)$ we can compute $z^{(t)}$, which is our output. □

The following circuit upper bound for the addressing (storage access) function is well-known (see, e.g., [Weg87]); we include a proof for completeness.

Lemma 14. *For any integers $t, m > 0$, there exists a circuit of size $O(tm)$ such that, given any string $y = (y_1, y_2, \ldots, y_t)$, where $y_i \in \{0,1\}^m$ for each i, and an index $i \in \{0,1\}^{\log t}$, the circuit outputs y_i.*

Proof. We first look at the first bit (i.e., the least significant bit in binary) of i and output either the first half of y (i.e., $y_1, y_2, \ldots, y_{t/2}$), if the first bit is 0, or the second half (i.e., $y_{(t/2)+1}, y_{(t/2)+2}, \ldots, y_t$), if the first bit is 1; denote this output as $y^{(1)}$. This can be done by a circuit of size $c \cdot t \cdot m$, for some constant $c > 0$. Then, we look at the second bit of i and output either the first half or the second half of $y^{(1)}$, denoted as $y^{(2)}$. This can be done by a circuit of size $c \cdot t \cdot m/2$. We repeat the above process $\log t$ times, in total, until we get $y^{(\log t)}$, which is y_i. The circuit complexity of this procedure is

$$\sum_{k=1}^{\log t} (c \cdot t \cdot m)/2^{k-1} = O(t \cdot m).$$ □

3 The “MCSP circuit lower bounds from local PRGs” framework

We first describe how to use local PRGs to obtain circuit lower bounds for MCSP.

Definition 15 (Local PRGs)**.** *Let $\lambda \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be a size function. For any Boolean computational model and size $s > 0$, we say that a function $G \colon \{0,1\}^{r=r(N,s)} \to \{0,1\}^N$ is a $(N, s, \lambda(N, s))$-local PRG against the model if*

- *G 1/3-fools every device f on N variables of size s in the model; that is,*

$$\left|\mathbb{E}_{z\sim\{0,1\}^r}[f(G(z))] - \mathbb{E}_{x\sim\{0,1\}^N}[f(x)]\right| \leq 1/3,$$

  and,

- *for any seed $z \in \{0,1\}^r$, the function $g \colon \{0,1\}^{\log N} \to \{0,1\}$ defined as $g_z(j) = G(z)_j$ can be computed by a general circuit of size at most $\lambda(N, s)$.*

Note that in the above definition, λ is at least as large as the seed length of the PRG.

Theorem 16. *There exists a constant $c > 0$ such that the following holds. For any computational model, let s be such that $\mathtt{MCSP}_N$ can be computed by a device of size s in the model. If there exists some $(N, s, \lambda(N, s))$-local PRG against the model, then $\lambda(N, s) \geq \frac{N}{c \log N}$.*

Proof. Let C be a device in the computational model such that C computes $\mathtt{MCSP}_N$. Suppose C has size s, and let G be a $(N, s, \lambda(N, s))$-local PRG against C with some seed length r.

For the sake of contradiction, suppose that

$$\lambda(N, s) < \frac{N}{c \log N}.$$

On the one hand, since most functions require circuits of size greater than $\frac{N}{c \log N}$ (Theorem 12) and C computes MCSP, we have

$$\mu := \mathbf{Pr}_{\mathtt{tt}(f)\sim\{0,1\}^N}[C(\mathtt{tt}(f)\,, \lambda(N, s)) = 0] \geq 1/2.$$

Also, since G fools C, we have

$$\mathbf{Pr}_{z\sim\{0,1\}^r}[C(G(z), \lambda(N, s)) = 0] \geq \mu - 1/3 \geq 1/6.$$

On the other hand, because G is $(N, s, \lambda(N, s))$-local, we must have

$$\mathtt{MCSP}(G(z), \lambda(N, s)) = 1,$$

for every z. A contradiction. □

It is easy to see that a local hitting set generator (HSG) is sufficient for the above argument to work. HSGs are a weak version of PRGs with the following property: For every function f in the class, if f accepts many of its inputs, then a HSG outputs such an input for at least one of its seeds.

4 Almost-cubic de Morgan formula lower bounds for MCSP

In this section, we present our almost-cubic de Morgan formula lower bound for MCSP. By saying "formula" within this section, we refer to formulas over the de Morgan basis (AND, OR, and NOT). By the size of a formula, we mean its usual leaf complexity, i.e., the number of leaves in the tree representation of the formula.

Theorem 17 (Theorem 1 restated)**.** *Any de Morgan formula computing* $\mathtt{MCSP}_N$ *has size at least* $N^3/2^{O\left(\log^{2/3} N\right)}$.

We will construct a strongly local PRG useful against sub-cubic formulas. That is, given as input an index j, the j-th bit of the PRG can be computed by a circuit of size that is comparable to its seed length, which in our case is around $s^{1/3}$ for size s formulas.

Lemma 18. *For any* $s \geq N$*, there exists a* $\left(N, s, s^{1/3} \cdot 2^{O\left(\log^{2/3} s\right)}\right)$*-local PRG against de Morgan formulas.*

Given the local PRG in Lemma 18, we can combine it with our Theorem 16 to obtain a formula lower bound for MCSP.

Proof of Theorem 17. Let s be such that $\mathtt{MCSP}_N$ can be computed by some formula of size s. We can assume $s \leq N^3$ since otherwise the result trivially holds. By Theorem 16 and Lemma 18, we have

$$s^{1/3} \cdot 2^{O\left(\log^{2/3} s\right)} \geq N/(c \log N);$$

then,

$$s \geq N^3 / \left(2^{O\left(\log^{2/3} N\right)} c^3 \log^3 N\right). \qquad \square$$

The rest of this section is devoted in proving Lemma 18.

4.1 Almost-linear-size extractors

Our PRG will make use of randomness extractors. Here, we describe an extractor that is computable by a circuit of size that is almost linear in the length of its input. We start by reviewing some basic definitions regarding extractors.

Definition 19 (ε-closeness and statistical distance)**.** *Let* $0 \leq \varepsilon \leq 1$*. We say two distributions* X *and* Y *(over some universe* D*) are* ε-close *if their statistical distance, defined as*

$$\max_{T: D \to \{0,1\}} \left| \mathbf{Pr}[T(X) = 1] - \mathbf{Pr}[T(Y) = 1] \right|,$$

is at most ε.

Definition 20 (Min-entropy)**.** *Let* X *be a random variable. The* min-entropy *of* X*, denoted by* $H_\infty(X)$*, is the largest real number* k *such that* $\mathbf{Pr}[X = x] \leq 2^{-k}$ *for every* x *in the range of* X*. If* X *is a distribution over* $\{0,1\}^{\aleph}$ *with* $H_\infty(X) \geq k$*, then* X *is called a* $(\aleph, k)$*-source.*

Definition 21 (Extractors)**.** *A function* $E\colon \{0,1\}^{\aleph} \times \{0,1\}^d \to \{0,1\}^m$ *is an* (k, ε)-extractor *if, for any* $(\aleph, k)$*-source* X*, the distribution* $E(X, U_d)$ *is* ε*-close to* U_m*.*

We now state the extractor, which for a high min-entropy source extracts a constant fraction of the min-entropy, using seeds of polylogarithimic length. The construction and circuit complexity of this extractor are presented in Appendix A.

Lemma 22 (Almost-linear-size extractors, following [NZ96])**.** *There exists some randomness extractor* $E\colon \{0,1\}^{\aleph} \times \{0,1\}^d \to \{0,1\}^m$ *that is an* $(\aleph/2, \varepsilon)$*-extractor with* $m = \Omega(\aleph)$ *and* $d = \mathsf{polylog}(\aleph/\varepsilon)$*. Moreover,* E *can be computed by a circuit of size* $\aleph \cdot \mathsf{polylog}(\aleph/\varepsilon)$*.*

4.2 Strongly local PRG useful against sub-cubic de Morgan formulas

For a formula F, let $L(F)$ denote the size (which is measured by the number of leaves) of F. We need the following pseudorandom shrinkage lemma for de Morgan formulas, which says that there exists a p-regular restriction, where the unrestricted variables are selected pseudorandomly and the restricted variables are fixed truly-randomly, such that *with high probability* the size of the restricted formula will "shrink" by a factor of p^2.

Lemma 23 (Pseudorandom shrinkage lemma, Lemma 4.8 of [IMZ12][4])**.** *There exists a constant* $c_0 > 0$ *such that the following holds. For any constant* $c > c_0$*, any* $s \geq N$*,* $p \geq s^{-1/2}$*, and any de Morgan formula* F *on* N *variables of size* s*, there exists a* p*-regular pseudorandom selection* $\mathcal{D}$ *over* N *variables that is samplable using* $r = 2^{O(\log^{2/3} s)}$ *random bits such that*

$$\mathbf{Pr}_{\sigma \sim \mathcal{D}, x \sim \{0,1\}^N} \left[L\big(F_{(\sigma,x)}\big) \geq 2^{3 \cdot c \cdot \log^{2/3} s} \cdot p^2 \cdot s \right] \leq s^{-c}.$$

Moreover, there exists a circuit of size $2^{O(\log^{2/3} s)}$ *such that, given* $j \in \{0,1\}^{\log N}$ *and a seed* $z \in \{0,1\}^r$*, the circuit computes the* j*-th bit of* $\mathcal{D}(z)$*.*

We are now ready to show our PRG in Lemma 18.

Proof of Lemma 18. The construction is as follows: We first sample a p-regular pseudorandom selection from Lemma 23. Then, we fill the star coordinates, specified by the pseudorandom selection, in the output string with the output of some extractor which takes a min-entropy source sample and a short seed (in fact, it is the output of some limited-independence generator that takes the output of the extractor as a seed). We then sample another pseudorandom selection, and fill the star coordinates specified by this pseudorandom selection but this time only for those that have not been filled in previous steps, again with the output of the same extractor using the *same min-entropy source sample* but a *different short seed*. We continue this way until all the coordinates are filled.

More formally, our PRG uses the following parameters:[5]

- $p = 1/s^{1/3}$, the expected fraction of unrestricted variables in each of the pseudorandom selections;
- $\varepsilon = 1/\mathsf{poly}(N)$ and $\varepsilon_0 = \varepsilon/(10t)$, which specify the error of the PRG;

[4] The pseudorandom shrinkage lemma in [IMZ12] is not stated in this form, but rather selects the unrestricted variables and fixes the restricted variables both pseudorandomly (based on limited independence). However, our version here follows from the proof of the original version in Section 4.2 of [IMZ12] by noting that limited-independence distributions can be computed locally.

[5] In fact, there are mainly two types of parameters here. Those that are close to $s^{1/3}$, which are $1/p, t, s_0, k, N$; and those that are close to $N^{o(1)}$, which are d and ℓ.

- $t = \ln(2N/\varepsilon)/p = s^{1/3} \cdot O(\log N)$, the number of steps needed so that all the coordinates will be filled with probability except $\varepsilon/2$;
- $s_0 = p^2 \cdot s \cdot 2^{O\left(\log^{2/3} s\right)} = s^{1/3} \cdot 2^{O\left(\log^{2/3} s\right)}$, the size of the formula after being simplified by a pseudorandom restriction;
- $k \geq s_0 = s^{1/3} \cdot 2^{O\left(\log^{2/3} s\right)}$, the amount of independence needed to fool the simplified formula, and $r_k = k \cdot \log N$ the seed length for the k-independent generator;
- $\aleph$, the length of the min-entropy source for the extractor, which is such that $\aleph \geq 2\cdot(\log(1/\varepsilon_0+c\cdot s_0\cdot\log s_0)$, where $c > 0$ is some constant, and that $\Omega(\aleph) \geq r_k$. We can take $\aleph = s^{1/3}\cdot 2^{O\left(\log^{2/3} s\right)}$;
- $d = \mathsf{polylog}(\aleph/\varepsilon_0) = \mathsf{polylog}(N)$, the seed length of the extractor;
- $\ell = 2^{O(\log^{2/3} s)}$, the number of random bits for sampling a pseudorandom selection.

Construction. The PRG takes a seed $(X, Y_1, Y_2, \ldots, Y_t, \gamma_1, \gamma_2, \ldots, \gamma_t) \in \{0,1\}^r$, where

- $X \in \{0,1\}^{\aleph}$ is the min-entropy source sample of an extractor,
- $Y_i \in \{0,1\}^{\mathsf{polylog}(N)}$, for each $i \in [t]$, is the seed of an extractor, and
- $\gamma_i \in \{0,1\}^{\ell}$, for each $i \in [t]$, is the seed for sampling a pseudorandom selection.

The construction of the PRG proceeds in the following two stages.

Stage 1: Compute a sequence of t p-regular pseudorandom selections

$$\sigma_1, \sigma_2, \ldots, \sigma_t,$$

using Lemma 23, with the seeds $\gamma_1, \gamma_2, \ldots, \gamma_t$. By abusing notation, we denote the star coordinates in σ_i by σ_i. Let $S_1, S_2, \ldots, S_t \subseteq [N]$ be t disjoint sets defined by

$$S_i = \sigma_i \setminus (S_1 \cup S_2 \cup \cdots \cup S_{i-1}).$$

Stage 2: Define $Z_1, Z_2, \ldots, Z_t \in \{0,1\}^N$ by

$$Z_i = G_k(E(X, Y_i)),$$

where $E\colon \{0,1\}^{\aleph} \times \{0,1\}^d \to \{0,1\}^{\Omega(\aleph)}$ is an $(\aleph/2, \varepsilon_0)$-extractor and $G_k\colon \{0,1\}^{r_k} \to \{0,1\}^N$ is a k-independent generator. The final output of our PRG is the binary string that has the values $Z_i|_{S_i}$ in the positions indexed by S_i, for all $i \in [t]$, where $Z_i|_{S_i}$ denotes the bit values of Z_i projected to the set S_i. (We fix those positions that are not in any of the S_i's to be 0.) Stage 2 of the PRG construction is depicted in Figure 1.

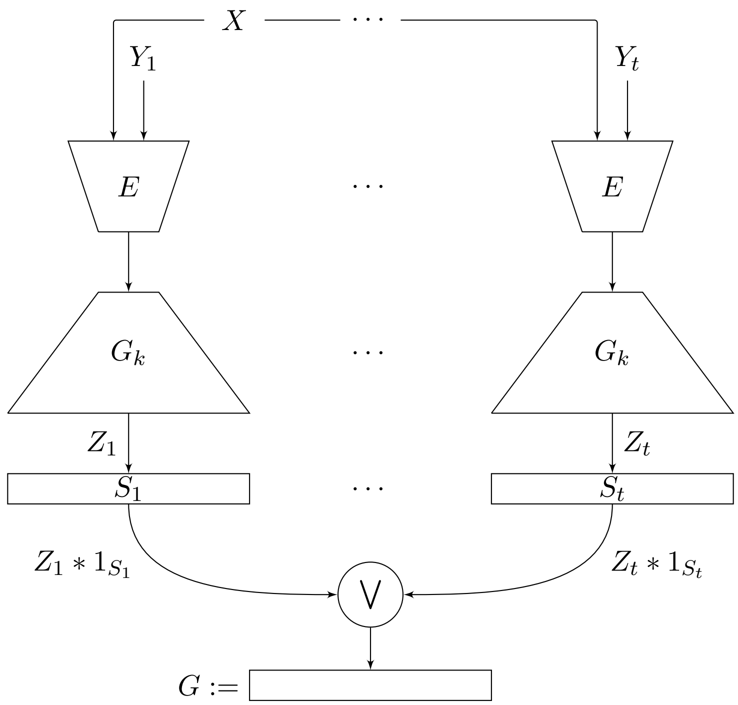

Figure 1: Construction of the PRG in Lemma 18, Stage 2. For each $i \in [t]$, $1_{S_i} \in \{0,1\}^N$ denotes the characteristic vector of the set S_i, where $S_i \subseteq [N]$ is the set of star coordinates in the i-th pseudorandom selection that did not appear in the preceding sets $S_1, \ldots, S_{i-1}$. Also, the operation $*$ denotes the coordinate-wise multiplication of vectors, and $\bigvee$ is the coordinate-wise OR operation.

Correctness. Next, we show that the above PRG ε-fools formulas of size s. First, note that, by our choice of t, with probability except $\varepsilon/2$, $S_1 \cup S_2 \cup \cdots \cup S_t$ covers all coordinates. For the rest of the argument, we will assume this is the case. By Lemma 10, conditioning on this assumption contributes at most $\varepsilon/2$ to the error of the constructed PRG.

We continue the correctness analysis using a *hybrid argument*. Let G denote the distribution given by the PRG described above. Let U be the uniform distribution. Note that if in the above construction we replace Z_i, for all $i \in [t]$, with U, then we would get a uniform distribution. Now we can start from there and we will gradually replace U with the Z_i's step-by-step for a total of t steps. We will argue that after each replacement step, the expected value of the function does not change by much. Let A_i be the distribution so that we have replaced U with Z_i in the first i steps. That is,

$$\begin{aligned} A_i &:= \left(Z_1|_{S_1}, Z_2|_{S_2}, \ldots, Z_i|_{S_i}, U|_{S_{i+1}}, \ldots, U|_{S_t} \right) \\ &= \left(Z_1|_{S_1}, Z_2|_{S_2}, \ldots, Z_i|_{S_i}, U|_{\overline{(S_{i+1} \cup \cdots \cup S_t)}} \right). \end{aligned}$$

For the sake of contradiction, suppose that

$$|\mathbb{E}[f(U)] - \mathbb{E}[f(G)]| = |\mathbb{E}[f(A_0)] - \mathbb{E}[f(A_t)]| > \varepsilon/2.$$

By the triangle inequality, there exists an $0 \le i < t$ such that

$$|\mathbb{E}[f(A_i)] - \mathbb{E}[f(A_{i+1})]| > \varepsilon/(2t). \tag{1}$$

Let us say that both the expectations in Equation (1) are over

$$\sigma_1, \sigma_2, \ldots, \sigma_i, \sigma_{i+1}, Y_1, Y_2, \ldots, Y_i, Y_{i+1}, X, U,$$

and we remove the absolute value without loss of generality. Then, we have

$$\mathbb{E}_{\substack{\sigma_1,\ldots,\sigma_i,\\ Y_1,\ldots,Y_i,\\ X}} \left[\mathbb{E}_{\sigma_{i+1},Y_{i+1},U}\left[f(A_i)\right] - \mathbb{E}_{\sigma_{i+1},Y_{i+1},U}\left[f(A_{i+1})\right]\right] > \varepsilon/(2t). \tag{2}$$

Denote $W_i = (\sigma_1, \ldots, \sigma_i, Y_1, \ldots, Y_i, X)$, and let f' be the random function (where the randomness is over W_i) defined as

$$f' := f(Z_1|_{S_1}, \ldots, Z_i|_{S_i}, \quad \cdot \quad).$$

That is, f' is the restricted function after the first i steps. Then, the left hand side of Equation (2) becomes

$$\mathbb{E}_{W_i}\left[\mathbb{E}_{\sigma_{i+1},U}\left[f'(U|_{S_{i+1}}, U|_{\overline{(S_{i+2}\cup\cdots\cup S_t)}})\right] - \mathbb{E}_{\sigma_{i+1},Y_{i+1},U}\left[f'(Z_{i+1}|_{S_{i+1}}, U|_{\overline{(S_{i+2}\cup\cdots\cup S_t)}})\right]\right]. \tag{3}$$

Note that at this point, we can view $\rho_{i+1} = (\sigma_{i+1}, U)$ as a pseudorandom restriction (in the sense of Lemma 23) applied to f'. Next, let f'' be the random function defined as the restricted function of f' under ρ_{i+1} (note that the randomness is over W_i, and also the pseudorandom restriction ρ_{i+1}). Now Equation (3) becomes

$$\mathbb{E}_{W_i,\rho_{i+1}}\left[\mathbb{E}_U\left[f''(U)\right] - \mathbb{E}_{Y_{i+1}}\left[f''(Z_{i+1})\right]\right]. \tag{4}$$

Note that in the above, we abuse the notation and use U and Z_{i+1} to denote $U|_{S_{i+1}}$ and $Z_{i+1}|_{S_{i+1}}$, respectively.

Next we want to show that the difference between the two expectations in Equation (4) is at most ε_0, which would give a contradiction. The intuition is the following. On the one hand, f'' is obtained by a pseudorandom restriction ρ_{i+1}, and so, with high probability, it has size at most s_0. On the other hand, Z_{i+1} is obtained using an extractor that is supposed to extract enough random bits for an s_0-independent generator.

The issue, however, is that f'' depends on X, the source sample of the extractor. Therefore, f'' may contain information about X, so that X is not truly random anymore. Nonetheless, being a formula of size at most s_0, f'' cannot contain too much information, and so cannot take too much entropy away from X. We make this argument more formal next.

Let us define the set of good functions for f'',

$$\mathcal{F} := \left\{g \mid L(g) \le s_0 \quad \text{and} \quad \mathbf{Pr}_{W_i,\rho_{i+1}}[f'' = g] \ge \varepsilon_0/s_0^{cs_0}\right\},$$

where c is some constant. Let $\mathcal{E}$ denote the event $f'' \in \mathcal{F}$. We first show the following.

Claim 24. *It is the case that* $\mathbf{Pr}[\neg\mathcal{E}] \le 2\varepsilon_0$.

Proof of Claim 24. We have

$$\begin{aligned}\mathbf{Pr}[\neg\mathcal{E}] &= \mathbf{Pr}[(f'' \notin \mathcal{F}) \wedge (L(f'') > s_0)] + \mathbf{Pr}[(f'' \notin \mathcal{F}) \wedge (L(f'') \le s_0)] \\ &\le \mathbf{Pr}[(L(f'') > s_0)] + \mathbf{Pr}[(f'' \notin \mathcal{F}) \wedge (L(f'') \le s_0)].\end{aligned}$$

Note that, by the pseudorandom shrinkage lemma (Lemma 23), we have

$$\mathbf{Pr}[L(f'') > s_0] \le \varepsilon_0.$$

Also note that under the condition that $L(f'') \le s_0$, there can be at most $s_0^{O(s_0)}$ choices for f'', since a formula of size s_0 can be specified using $O(s_0 \log s_0)$ bits (Proposition 11). Therefore,

$$\mathbf{Pr}\big[(f'' \notin \mathcal{F}) \wedge (L(f'') \le s_0)\big] \le s_0^{O(s_0)} \cdot \varepsilon_0 / s_0^{cs_0} \le \varepsilon_0. \qquad \square$$

Let us now analyze Equation (4) while conditioning on the event $\mathcal{E}$. We show the following.

Claim 25. *It is the case that* $\mathbb{E}\left[f''(U) \mid \mathcal{E}\right] - \mathbb{E}\left[f''(Z_{i+1}) \mid \mathcal{E}\right] \le \varepsilon_0$.

Proof of Claim 25. First note that conditioning on $\mathcal{E}$, X still has a large min-entropy. More precisely, for every $g \in \mathcal{F}$ it is the case that

$$H_\infty\big(X \mid f'' = g\big) \ge \aleph/2.$$

This is because, for every x, we have

$$\mathbf{Pr}[X = x \mid f'' = g] \le \frac{\mathbf{Pr}[X = x]}{\mathbf{Pr}[f'' = g]} \le \frac{2^{-\aleph}}{\varepsilon_0 / s_0^{c \cdot s_0}} = 2^{-(\aleph - \log(1/\varepsilon_0) - c \cdot s_0 \cdot \log s_0)} \le 2^{-\aleph/2}.$$

Then, by the definition of the extractor, we have

$$\mathbb{E}\left[f''(G_k(U)) \mid \mathcal{E}\right] - \mathbb{E}\left[f''(Z_{i+1}) \mid \mathcal{E}\right] \le \varepsilon_0.$$

Finally, note that

$$\mathbb{E}\left[f''(G_k(U)) \mid \mathcal{E}\right] = \mathbb{E}\left[f''(U) \mid \mathcal{E}\right],$$

since s_0-wise independent distributions fool size-s_0 formulas. $\square$

Combining Claim 24, Claim 25, and Lemma 10, we get that the quantity in Equation (4) is at most $3\varepsilon_0$, which leads to a contradiction. This completes the proof of the correctness.

Locality. To see that the j-th bit of the PRG can be computed using a circuit of size $s^{1/3} \cdot 2^{O\left(\log^{2/3} s\right)}$, we observe the following equivalent construction:

1. Compute the j-th bits of the t pseudorandom selections $(\sigma_1)_j, (\sigma_2)_j, \ldots, (\sigma_t)_j$.
2. Retrieve Y_q, where q is the smallest integer such that $(\sigma_q)_j$ is a star.
3. Compute $(Z_q)_j = G_k(E(X, Y_q))_j$ as the j-th bit of the PRG.

Note that Step 1 can be done using a circuit of size $t \cdot 2^{O\left(\log^{2/3} s\right)} = s^{1/3} \cdot 2^{O\left(\log^{2/3} s\right)}$, by the pseudorandom shrinkage lemma (Lemma 23). Also, Step 2 can be done by first computing q from the sequence $((\sigma_i)_j)_{i \in [t]}$ using a circuit of size $\widetilde{O}(t)$ (Lemma 13), and then outputting Y_q from $(Y_i)_{i \in [t]}$ using a circuit of size $t \cdot \mathsf{polylog}(N)$ (Lemma 14). Finally, Step 3 can be done by a circuit of size $\widetilde{O}(\aleph)$ using the efficient extractor (Lemma 22) and the limited-independence generator (Lemma 8). $\square$

5 Almost-quadratic lower bounds against arbitrary basis formulas and branching programs

Here, we prove MCSP lower bounds against formulas over an arbitrary basis, and branching programs. These lower bounds are obtained similarly to those for de Morgan formulas in the previous section. The idea is to construct strongly local PRGs against these models by modifying the PRGs in [IMZ12].

The following pseudorandom shrinkage lemma for formulas over an arbitrary basis as well as branching programs is an analogue of Lemma 23.

Lemma 26 (Lemma 4.2 and Lemma 5.3 of [IMZ12])**.** *There exists a constant $c_0 > 0$ such that the following holds. For any constant $c > c_0$ and any $s \geq N$, let $p = s^{-1/2}$ and a formula F over any basis (or a branching program) on N variables of size s, there exists a p-regular pseudorandom selection $\mathcal{D}$ over N variables that is samplable using $r = \mathsf{polylog}(N)$ random bits such that*

$$\mathbf{Pr}_{\sigma\sim\mathcal{D},x\sim\{0,1\}^N}\left[L\big(F_{(\sigma,x)}\big) \geq 2^{3\cdot\sqrt{c\cdot\log s}}\cdot p\cdot s\right] \leq 2\cdot s^{-c}.$$

Moreover, there exists a circuit of size $2^{O(\log^{2/3} s)}$ such that, given $j \in \{0,1\}^{\log N}$ and a seed $z \in \{0,1\}^r$, the circuit computes the j-th bit of $\mathcal{D}(z)$.

Using the above pseudorandom shrinkage lemma and an argument as in the proof of the strongly local PRG against de Morgan formulas (Lemma 18), we get the following local PRGs.

Lemma 27. *For any $s \geq n$, there exists a $\left(N, s, s^{1/2}\cdot 2^{O\left(\sqrt{\log s}\right)}\right)$-local PRG against size s formulas over an arbitrary basis (or branching programs).*

The MCSP lower bound in Theorem 2 follows from Lemma 27 and Theorem 16.

6 Improved AC^0 lower bounds for MCSP

In this section, we show improved lower bounds for MCSP against constant-depth circuits.

6.1 The case of depth $d > 2$

We first show an improved lower bound against depth-d circuits that almost matches the lower bound for PARITY.

Theorem 28 (Theorem 3 restated)**.** *For every $d > 2$ and every constant $\gamma > 0$, any depth-d AC^0 circuit computing $\mathtt{MCSP}_N$ must have size $2^{\Omega\left(N^{1/(d+2+\gamma)}\right)}$.*

The above result is proved using the following structural property of small-depth circuits, which says that, for any such circuit, there exists some locally computable restriction that simplifies the circuits to a constant while leaving many variables unrestricted.

Lemma 29. *For any size-s depth-d circuit C, there exists a restriction $\rho \in \{0,1,*\}^N$ such that*

- *C_ρ is a constant function,*
- *$\left|\rho^{-1}(*)\right| \geq \frac{N}{O(\log s)^{d-2}} - \log s$, and*

- *there exists a circuit of size $d \cdot \log(N) \cdot \widetilde{O}\left(\log^3 s\right)$ such that, given $j \in \{0,1\}^{\log N}$, the circuit computes the j-th coordinate of ρ.*

The proof of Lemma 29 uses the pseudorandom switching lemma due to Trevisan and Xue [TX13], which we revisit below. The (pseudorandom) switching lemma says that a depth-2 circuit is likely to be simplified after hit by a (pseudo)random restriction.

Lemma 30 (Pseudorandom switching lemma, Lemma 7 of [TX13])**.** *For any integers $t, w > 0$, $s \geq N$, and any $0 < p, \varepsilon_0 < 1$, let F be a N-variate w-CNF or w-DNF of size s, and let $\mathcal{D}$ be a distribution over $\{0,1\}^{N \cdot \log(1/p)} \times \{0,1\}^N$ that ε_0-fools $\left(s_0 = s \cdot 2^{w \cdot (\log(1/p)+1)}\right)$-clause CNFs, then*

$$\mathbf{Pr}_{\rho \sim \mathcal{D}}[F_\rho \textit{ does not have a depth-}t\textit{ decision tree}] \leq 2^{t+w+1} \cdot (5 \cdot p \cdot w)^w + \varepsilon_0 \cdot 2^{(t+1)(2w+\log s)}.$$

Lemma 31 (Following Theorem 11 of [TX13])**.** *For any integers $d, t > 0$, $s \geq N$, and any $1/n < p < 1$ and $0 < \varepsilon_0 < 1$, there exists a distribution $\mathcal{D}$ over $\{0,1\}^{N \cdot \log(1/p)} \times \{0,1\}^N$ for sampling a pseudorandom restriction such that*

- *for any size-s depth-d circuit C on N variables, we have that*

$$\mathbf{Pr}_{\rho \sim \mathcal{D}}\left[C_\rho \textit{ is not a } t\text{-}\mathsf{DNF} \textit{ or } t\text{-}\mathsf{CNF}\right] \leq s \cdot \left(2^{2t+1} \cdot (10 \cdot p \cdot \log s)^t + \varepsilon_0 \cdot 2^{(t+1)(2t+\log s)}\right),$$

- *with probability at least $2/3$ the number of unrestricted variables is $\frac{p^{d-2}}{80} \cdot N$, and*
- *there exists a circuit of size $d \cdot k \cdot \widetilde{O}(\log N)$ such that, given $j \in \{0,1\}^{\log N}$ and a seed $z \in \{0,1\}^{d \cdot k \cdot O(\log N)}$, the circuit computes the j-th coordinate of ρ (as an element in $\{0,1,*\}$), where*

$$k = O\left((\log(s) + t \cdot \log(1/p))^2 + (\log(s) + t \cdot \log(1/p)) \cdot \log(1/\epsilon_0)\right).$$

Proof (sketch). The proof is similar to that of Theorem 11 in [TX13]. The idea is to apply the pseudorandom switching lemma (Lemma 30) repeatedly. Each time, we sample a pseudorandom restriction using some distribution that ε_0-fools CNF of size $s_0 = s \cdot 2^{w \cdot (\log(1/p)+1)}$ for $w = t$. By Lemma 30, each time, with high probability, the bottom two layers can be computed by depth-t decision trees, so we can switch them to t-DNFs or t-CNFs, and hence reduce the depth of the circuit by one as we merge them with the layer above.

One difference here from the argument in [TX13] is that we only apply the pseudorandom switching lemma $d-1$ times, instead of d times, since we only need the final restricted circuit to be a t-DNF or t-CNF (rather than a depth-t decision tree as in the original statement of [TX13], which requires an additional application of the pseudorandom switching lemma). Note that we use parameter $p = 1/40$ for the first iteration. Another difference is that, to sample a pseudorandom restriction, we use a k-wise independent distribution (say over $[1/p]^{2N}$), instead of using the PRG against depth-2 circuits in [DETT10], where

$$k = O\left(\log(s_0/\varepsilon_0) \cdot \log s_0\right) = O\left((\log(s) + t \cdot \log(1/p))^2 + (\log(s) + t \cdot \log(1/p)) \cdot \log(1/\epsilon_0)\right),$$

and we use the fact that such a k-wise independent distribution ε_0-fools s_0-clause CNFs [Tal17b].[6]

Note that Item 2 follows from Proposition 9 and the fact that the expected number of unrestricted variables is $\frac{p^{d-2}}{40} \cdot N$. Finally, it is easy to get Item 3 using Lemma 8. $\square$

[6] The PRG in [DETT10] is based on a *small-biased distribution*. While it has a smaller seed length, compared to a k-wise independent distribution, it does not seem to offer any advantage in terms of the local circuit complexity of computing the PRG.

We are now ready to show Lemma 29.

Proof of Lemma 29. By Lemma 31, using the parameters $t = O(\log s)$, $p = 1/O(\log s)$, and $\varepsilon_0 = 1/2^{O\left(\log^2 s\right)}$, we get a restriction ρ_0 such that the circuit restricted by ρ_0 is a width-$O(\log s)$ DNF or CNF, with probability at least $1 - 1/\mathsf{poly}(N)$. Note that by Item 2 of Lemma 31, ρ_0 leaves at least $\frac{N}{O(\log s)^{d-2}}$ variables unrestricted with constant probability. Therefore, by the union bound, with some constant probability, we get a restriction ρ_0 that simplifies the circuit to be a width-$(\log s)$ DNF or CNF and that leaves $\frac{N}{O(\log s)^{d-2}}$ variables unrestricted. Note that once we have such a restriction, we can make the restricted circuit constant by further fixing at most $\log s$ variables; denote this restriction by ρ_1. The final restriction is $\rho := \rho_0 \circ \rho_1$.

We now show the last item. Note that our final restriction consists of two parts, ρ_0 and ρ_1, where ρ_0 is a restriction from Lemma 31 and ρ_1 is a restriction that fixes $\log s$ variables. To compute the final restriction, given an index $j \in \{0,1\}^{\log N}$, we can first check if the j-th variable is fixed by ρ_1 and output the fixing value if it is the case. This can be done by hard wiring the $\log s$ variables that are fixed by ρ_1 and their corresponding fixing values. It is easy to see that the above can be done using a circuit of size at most $O(\log s \cdot \log N)$. Otherwise, we can output the j-th coordinate of ρ_0, which can be done with a circuit of size $d \cdot \log(N) \cdot \widetilde{O}\left(\log^3 s\right)$ by Item 3 of Lemma 31. $\square$

We now prove Theorem 28 using Lemma 29.

Proof of Theorem 28. Let C be a depth-d AC^0 circuit on $\{0,1\}^N \times \{0,1\}^{\log N}$ such that C computes $\mathtt{MCSP}_N$, and let s be the size of C.

For a size parameter $\lambda = d \cdot \log(N) \cdot \widetilde{O}\left(\log^3 s\right)$, let $C' = C(\cdot, \lambda)$. Let ρ be a restriction from Lemma 29 for C'. By Lemma 29, we have that C'_ρ is a constant function. First, note that

$$C'_\rho\left(0^{|\rho^{-1}(*)|}\right) = 1.$$

To see this, note that

$$C'_\rho\left(0^{|\rho^{-1}(*)|}\right) = C\left(\mathtt{tt}(f), \lambda\right),$$

where C computes $\mathtt{MCSP}$ and $f\colon \{0,1\}^{\log N} \to \{0,1\}$ is the following:

$$f(j) = \begin{cases} 0, & \text{if } \rho_j = 0 \text{ or } \rho_j = *, \\ 1, & \text{if } \rho_j = 1. \end{cases}$$

By Item 3 of Lemma 29, such a function f can be computed by a λ-size circuit. On the other hand, there can be $2^{|\rho^{-1}(*)|}$ different functions corresponding to the different partial assignments to the unrestricted variables. Since there are at most $2^{O(\lambda \log \lambda)}$ different circuits of size at most λ, in order for C'_ρ to be the constant 1, we must have

$$2^{O(\lambda \log \lambda)} \geq 2^{|\rho^{-1}(*)|} = 2^{\frac{N}{O(\log s)^{d-2}} - \log s},$$

which, by a simple calculation, implies $s = 2^{\Omega\left(N^{1/(d+2+\gamma)}\right)}$ for any constant $\gamma > 0$. $\square$

6.2 The case of depth 2

Here, we show that computing MCSP requires depth-2 circuits of almost maximum size.

Theorem 32 (Theorem 4 restated)**.** *Any* CNF *or* DNF *computing* MCSP_N *must have size* $2^{N/\widetilde{O}\left(\log^2 N\right)}$.

The proof uses the following variant of Lemma 29 which says that a depth-2 circuit can be made constant via a more efficient restriction.

Lemma 33. *For any size-s depth-2 circuit C, there exists a restriction $\rho \in \{0,1,*\}^N$ such that*

- *C_ρ is a constant function,*
- *$\left|\rho^{-1}(*)\right| \geq \Omega(N) - \log s$, and*
- *there exists a circuit of size $\log(s) \cdot \widetilde{O}(\log N)$ such that, given $j \in \{0,1\}^{\log N}$, the circuit computes the j-th coordinate of ρ.*

Given Lemma 33, it is straightforward to prove Theorem 32 following the argument in the proof of Theorem 28. We omit the details here.

Next, we show Lemma 33. The proof uses the following observation which says that a small CNF or DNF is likely to have small width after being hit by a mild pseudorandom restriction.

Lemma 34. *For any positive integer w and any $1/N < p < 1/16$, let $\mathcal{R} = (\sigma, \beta)$ be a random restriction such that*

- *σ is sampled using a $(w \cdot \log(1/p))$-wise independent distribution over $[1/p]^N$ and*
- *β is sampled using a $(w \cdot \log(1/p))$-wise independent distribution over $\{0,1\}^N$.*

Then, for any size-s CNF *or* DNF *C on N variables, we have*

$$\mathbf{Pr}_{\rho\sim\mathcal{R}}[\textit{the width of } C_\rho \textit{ is greater than } w] \leq s \cdot (3 \cdot p \cdot \log(1/p))^w.$$

Proof. Assume without loss of generality that C is a CNF. Let M be any clause in C. Suppose $|M| \leq w \cdot \log(1/p)$. Then,

$$\begin{aligned}
\mathbf{Pr}_{\rho\sim\mathcal{R}}[|M_\rho| \geq w] &\leq \sum_{S\subseteq M:|S|=w} \mathbf{Pr}_\rho[\text{all variables in } S \text{ are left unrestricted by } \rho] \\
&= \sum_{S\subseteq M:|S|=w} p^w \\
&= \binom{|M|}{w} \cdot p^w \\
&\leq \binom{w \cdot \log(1/p)}{w} \cdot p^w \\
&\leq (e \cdot \log(1/p))^w \cdot p^w \\
&\leq (3 \cdot p \cdot \log(1/p))^w.
\end{aligned}$$

In the above, we used the fact that $\mathcal{R}$ picks the set of unrestricted variables w-wise independently, each with probability p.

Now suppose that $|M| \geq w \cdot \log(1/p)$. Let S be any subset of M such that $|S| = w \cdot \log(1/p)$. Then,

$$\begin{aligned}
\mathbf{Pr}_{\rho\sim\mathcal{R}}[|M_\rho| \geq w] &\leq \mathbf{Pr}_\rho[M_\rho \neq 0] \\
&= \mathbf{Pr}_\rho[\text{no literal in } S \text{ is set to 0 by } \rho] \\
&= \left(1 - \frac{1-p}{2}\right)^{|S|} \\
&\leq \left(\frac{17}{32}\right)^{w\cdot\log(1/p)} \\
&\leq (3 \cdot p \cdot \log(1/p))^w.
\end{aligned}$$

In the above, we made use of the fact that any $w \cdot \log(1/p)$ coordinates of $\mathcal{R}$ are independent. The lemma then follows by applying the union bound over all s clauses. $\square$

We now prove Lemma 33.

Proof of Lemma 33. Consider the random restriction in Lemma 34 that uses limited independence, with the parameters $w = \log s$ and $p = 1/64$. We have, for every CNF or DNF C, with high probability, a restriction under which C has width at most w. Also, since the expected number of unrestricted variables is $N/64$, by Proposition 9, with constant probability it leaves $\Omega(N)$ variables unrestricted. Therefore, by the union bound, there exists some restriction ρ_0 that satisfies both conditions. Once we have such a restriction, we can combine it with another restriction ρ_1 that fixes at most w variables to obtain a restriction $\rho = \rho_0 \circ \rho_1$ such that C_ρ is a constant. Also, the number of unrestricted variables by ρ is $\Omega(N) - \log s$. For the last item, note that ρ_0, which is sampled using a $(w \cdot (1/p))$-wise independent distribution (say over $[1/p]^{2N}$), can be computed by a circuit of size $\log(s) \cdot \widetilde{O}(\log N)$ (Lemma 8). Also, it is easy to see that by hard-wiring the w restricted variables and their fixed values, ρ_1 can be computed by a circuit of size $\log(s) \cdot O(\log N)$. As a result, we can compute ρ using a circuit of size $\log(s) \cdot \widetilde{O}(\log N)$. $\square$

7 MCSP circuit lower bounds from average-case hard functions

7.1 The Nisan-Wigderson Generator

It is well known in the field of derandomization that, if we have a function that is average-case hard against some circuit class $\mathfrak{C}$, we can get a PRG for $\mathfrak{C}$ by plugging the hard function into the Nisan-Wigderson framework [NW94] (provided that the hard function is not too hard to compute and that $\mathfrak{C}$ satisfies some mild conditions). The construction involves computing some combinatorial design with some suitably chosen parameters; a design is a list of subsets (over some universe) that have some combinatorial properties (see Definition 35). Also, to compute a single bit of such a PRG, we need to compute the corresponding subset of the design. There are known design constructions such that any single subset of the design can be computed efficiently and locally (without computing the whole design). Therefore, using such a local design, we can get a locally computable PRG which can be used to obtain an MCSP lower bound against $\mathfrak{C}$.

The idea of using Nisan-Wigderson PRGs to study MCSP and related problems has been explored before (e.g. [ABK+06, OS17, Hir18]). However, the previous works were content with the fact that the output of a PRG has the circuit complexity at most polynomial in the seed length. Here, we provide a more fine-grained analysis of the local complexity of the Nisan-Wigderson PRG, which depends on the parameters that we choose for the design, and in turn will depend on the "usefulness" of the average-case hard function. This allows us to turn average-case hardness against some circuit class $\mathfrak{C}$ into a lower bound for MCSP against the same class, where such a lower bound is more quantitatively linked to the average-case hardness.

We first review the Nisan-Wigderson framework.

Definition 35 (Designs [NW94]). *Let N, r, ℓ, α be positive integers. A family of sets $S_1, S_2, \ldots, S_N$ is a (N, r, ℓ, α)-design if*

- $\forall j \in [N] : S_j \subseteq [r]$,
- $\forall j \in [N] : |S_j| = \ell$, *and*
- $\forall j, k \in [s]$, *such that $j \neq k$, it is the case that* $|S_j \cap S_k| \leq \alpha$.

Lemma 36 (Local design). *For any positive integers N and α, there exists a (N, r, ℓ, α)-design such that $r = N^{2/(\alpha+1)}$ and $\ell = N^{1/(\alpha+1)}$. Moreover, given any $z \in \{0,1\}^r$, and any $j \in \{0,1\}^{\log N}$, $z|_{S_j} \in \{0,1\}^\ell$ can be computed by a circuit of size*

$$O\left(N^{2/(\alpha+1)}\right) + N^{1/(\alpha+1)} \cdot \widetilde{O}(\log N).$$

Proof. Consider the field $\mathbb{F}_\ell$ with ℓ elements. We identify the universe $[r]$ with $\mathbb{F}_\ell \times \mathbb{F}_\ell$ of size ℓ^2. Let $\{e_1, e_2, \ldots, e_\ell\}$ be the ℓ elements of the field (in lexicographic order). For each $j \in \{0,1\}^{\log N}$, we view j as an element in $[\ell]^{\alpha+1}$ and identify it with a degree-α polynomial $p_j \in \mathbb{F}_\ell[x]$. Let

$$S_j = \{(e_1, p_j(e_1)), (e_2, p_j(e_2)), \ldots, (e_\ell, p_j(e_\ell))\}.$$

Note that, for all j, the set S_j is a subset of $\mathbb{F}_\ell \times \mathbb{F}_\ell$, the set S_j has size ℓ, and for two different sets S_j and S_k we have that $|S_j \cap S_k| \leq \alpha$, as the difference $p_j - p_k$ is a polynomial of degree at most α, and thus has at most α roots.

Note that we can hard wire $\left(e_k, e_k^2, \ldots, e_k^\alpha\right)$ into some circuit, for all $k \in [\ell]$. Then computing $p_j(e_k)$, for any k, can be done with a circuit of size $\alpha \cdot \widetilde{O}(\log \ell)$ (using Fact 7). As a result, S_j can be computed in size

$$\ell \cdot \alpha \cdot \widetilde{O}(\log \ell) = N^{1/(\alpha+1)} \cdot \widetilde{O}(\log N).$$

Once we have the set S_j, we can divide the input z into ℓ equal-size blocks. For each element (a, b) in S_j, we output the b-th bit of the a-th block, using Lemma 14, in $O(\ell)$ size. Then, computing $z|_{S_j}$ takes size $\ell \cdot O(\ell) = O\left(N^{2/(\alpha+1)}\right)$. ☐

Definition 37 (Average-case hard). *Let $\mathfrak{C}$ be a class of circuits on N variables. We say that a function f is (s, ε)-hard against $\mathfrak{C}$ if, for every $C \in \mathfrak{C}$, it is the case that*

$$\mathbf{Pr}_{x \sim \{0,1\}^N}[f(x) = C(x)] \leq \frac{1}{2} + \varepsilon.$$

Let DNF_α denote the class of DNF circuits on α variables. Note that every α-variate Boolean function can be computed by a DNF of size at most 2^α.

Theorem 38 (Nisan-Wigderson generator [NW94])**.** *Let $\mathfrak{C}$ be a class of circuits on N variables of size s. Let $S_1, S_2, \ldots, S_N$ be a (N, r, ℓ, α)-design, and let $f : \{0,1\}^\ell \to \{0,1\}$ be a function that is $(s + N \cdot 2^\alpha, \varepsilon/N)$-hard against $\mathfrak{C} \circ \mathsf{DNF}_\alpha$. Then, the Nisan-Wigderson generator* $\mathrm{NW}^f : \{0,1\}^r \to \{0,1\}^N$, *defined as*

$$\mathrm{NW}^f(z) = \left(f\left(z|_{S_1}\right), f\left(z|_{S_2}\right), \ldots, f\left(z|_{S_s}\right)\right),$$

is a PRG that ε-fools $\mathfrak{C}$.

Combining Theorem 38 with the design construction in Lemma 36, we immediately get the following.

Theorem 39 (Local Nisan-Wigderson generator)**.** *Let $\mathfrak{C}$ be a class of circuits on N variables of size s. For any $\alpha = \alpha(N, s)$, if there exists a function $f\colon \{0,1\}^\ell \to \{0,1\}$, where $\ell = N^{1/(\alpha+1)}$, that is $(s + N \cdot 2^\alpha, 1/(3N))$-hard against $\mathfrak{C} \circ \mathsf{DNF}_\alpha$, then there exists a $(N, s, \lambda(N, s))$-local PRG against $\mathfrak{C}$, with*

$$\lambda(N, s) = O\left(N^{2/(\alpha+1)}\right) + N^{1/(\alpha+1)} \cdot \widetilde{O}(\log N) + \mathtt{CC}(f)\,.$$

We remark that the above local Nisan-Wigderson generator has local complexity that is comparable to its seed length (for this particular local design and modulo the circuit complexity of the hard function).

7.2 Applications

Next we demonstrate the use of such local PRGs in obtaining lower bounds for $\mathtt{MCSP}$ from average-case hardness results.

One of the restricted circuit classes that have been well studied in circuit complexity is the class of constant-depth circuits augmented with few SYM (symmetric) or THR (linear threshold) gates (see, e.g., [LVW93, Vio07, LS11, ST18]). A SYM gate computes a symmetric function, which is a Boolean function whose output depends only on the sum of its input variables. A THR gate computes a linear threshold function, which is a Boolean function defined as the sign of some linear form, on Boolean variables, with real coefficients. We will combine the above local Nisan-Wigderson framework with the following average-case lower bounds against the class of constant-depth circuits augmented with a few (sublinearly many) symmetric and linear threshold gates.

Theorem 40 (Theorem 3 of [ST18])**.** *There exists a constant $\tau > 0$ such that the following hold. For any ℓ, there exists a function $f\colon \{0,1\}^\ell \to \{0,1\}$ that is $\left(\ell^{\tau \log \ell}, \exp\left(-\Omega(\ell^{0.499})\right)\right)$-hard against $\{\mathsf{SYM}, \mathsf{THR}\} \circ \mathsf{AC}^0$. Moreover f can be computed by a circuit of size $O(\ell)$.*

As a result, we get a local PRG against such circuits.

Corollary 41. *For any $s \geq N$, there exists a $(N, s, \lambda(N, s))$-local PRG against $\{\mathsf{SYM}, \mathsf{THR}\} \circ \mathsf{AC}^0$, with $\lambda(N, s) = 2^{O(\sqrt{\log s})}$.*

Proof. Choose

$$\alpha = \tau' \cdot \frac{\log N}{\sqrt{\log s} + \log\log N},$$

where $\tau' > 0$ is some sufficiently small constant. Then, for $\ell = N^{1/(\alpha+1)}$, if we can show the existence of some efficiently computable function $f\colon \{0,1\}^\ell \to \{0,1\}$ that is $(s + N \cdot 2^\alpha, 1/(3N))$-hard against

$\{\mathsf{SYM}, \mathsf{THR}\} \circ \mathsf{AC}^0 \circ \mathsf{DNF}_\alpha$, then the result follows from Theorem 39. The existence of such a function is given by Theorem 40, by noting that for our choice of α we have

$$\ell^{\tau \log \ell} \geq s + N \cdot 2^\alpha,$$

and

$$\exp\left(-\Omega(\ell^{0.499})\right) \leq 1/(3N). \qquad \square$$

We remark that the above example does not take advantage of the fact that the local complexity of the Nisan-Wigderson PRG is almost the same as its seed length. This is because, in this case, the seed length has some arbitrary constant in the exponent.

Combining Corollary 41 with Theorem 16, we get the following.

Theorem 42. *Any circuit in $\{\mathsf{SYM}, \mathsf{THR}\} \circ \mathsf{AC}^0$ computing $\mathtt{MCSP}_N$ must have size $N^{\Omega(\log N)}$.*

By a similar argument, and using the function f on ℓ variables that is average-case hard against the class of AC^0 circuits augmented with at most $\ell^{0.249}$ SYM or THR gates of size $\ell^{\tau \cdot \sqrt{\log \ell}}$, given by [ST18, Theorem 14], we can get the following.

Theorem 43. *There exists a constant $\gamma > 0$ such that the following holds. Let $\mathfrak{C}$ be the class of constant-depth AC^0 circuits augmented with at most $2^{\gamma\sqrt{\log N}}$ SYM or THR gates. Then any circuit in $\mathfrak{C}$ computing $\mathtt{MCSP}_N$ must have size $N^{\Omega\left(\sqrt{\log N}\right)}$.*

As another application of our framework, combined with the Nisan-Wigderson generator, we show that separating P/poly (non-uniform circuits of polynomial size) from some restricted circuit class, such as TC^0 (non-uniform constant-depth polynomial-size circuits with threshold gates) or NC^1 (non-uniform polynomial-size logarithmic-depth circuits), implies a function in NP that is hard against the same class of circuits. More precisely, we show that if there exists some function in P/poly that is mildly hard against TC^0 (resp. NC^1), then $\mathtt{MCSP}$ cannot be computed by TC^0 (resp. NC^1) circuits.

Theorem 44. *If there exists a function in P/poly that requires size-s TC^0 (resp. NC^1) circuits to compute within error $1/\mathsf{poly}(n)$, for some superpolynomial size s, then NP requires superpolynomial size TC^0 (resp. NC^1) circuits.*

Proof (sketch). Let $s(n) = n^{\omega(1)}$, and let $f = \{f_n\}_n$, with $f_n \colon \{0,1\}^n \to \{0,1\}$, be a function that requires size-$s(n)$ TC^0 circuits to compute with error at most $1/\mathsf{poly}(n)$. Using standard hardness amplification tools, such as the direct product theorem and the XOR lemma (see, e.g., Section 4 of [CIKK16]), we can amplify f to a strongly hard on average function within P/poly. By plugging f into the Nisan-Wigderson construction (Theorem 38) we get a local PRG against TC^0; this implies that $\mathtt{MCSP} \notin \mathsf{TC}^0$ by Theorem 16. $\square$

8 Open problems

Our de Morgan formula lower bound for $\mathtt{MCSP}$ is still slightly weaker than the state-of-the-art de Morgan formula lower bound due to Tal [Tal17a], which is $\Omega\big(N^3/\left(\log N \cdot (\log\log N)^2\right)\big)$. Can the MCSP lower bound be improved? Are there better constructions of local PRGs against formulas? Or, are there alternative proofs that do not rely on local PRGs?

A similar question can be asked for small-depth circuits. In particular, can we show that MCSP requires depth-2 circuits (i.e., CNFs or DNFs) of size $2^{\Omega(N)}$, as in the case of PARITY?

What are other restricted models of computation against which we can show MCSP lower bounds using local PRGs? The recent "random walk PRG" by Chattopadhyay, Hatami, Hosseini, and Lovett [CHHL18] is also local and can be used to get MCSP lower bounds. However, as a general PRG that can be used to fool a variety of restricted models, it has sub-optimal usefulness (which is determined by the seed length) compared to the best-known lower bounds for most of those 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. doi:10.1137/050628994. 3, 4, 21

[AW89] Miklós Ajtai and Avi Wigderson. Deterministic simulation of probabilistic constant depth circuits. *Advances in Computing Research*, 5:199–222, 1989. doi:10.1109/SFCS.1985.19. 5

[CHHL18] Eshan Chattopadhyay, Pooya Hatami, Kaave Hosseini, and Shachar Lovett. Pseudorandom generators from polarizing random walks. In *CCC*, pages 1:1–1:21, 2018. ECCC:TR18-015. 25

[CIKK16] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In *CCC*, pages 10:1–10:24, 2016. ECCC:TR16-008. 24

[DETT10] Anindya De, Omid Etesami, Luca Trevisan, and Madhur Tulsiani. Improved pseudorandom generators for depth 2 circuits. In *APPROX/RANDOM*, pages 504–517, 2010. ECCC:TR09-141. 18

[DM18] Irit Dinur and Or Meir. Toward the KRW composition conjecture: Cubic formula lower bounds via communication complexity. *Computational Complexity*, 27(3):375–462, 2018. doi:10.1007/s00037-017-0159-x. 3

[GS13] Sergey B. Gashkov and Igor S. Sergeev. Complexity of computation in finite fields. *Journal of Mathematical Sciences*, 191(5):661–685, 2013. doi:10.1007/s10958-013-1350-5. 7

[Hås86] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *STOC*, 1986. doi:10.1145/12130.12132. 3

[Hås98] Johan Håstad. The shrinkage exponent of de Morgan formulas is 2. *SIAM J. Comput.*, 27(1):48–64, 1998. doi:10.1137/S0097539794261556. 3

[Hir18] Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *FOCS*, pages 247–258, 2018. ECCC:TR18-136. 4, 21

[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. 3, 4

[IMZ12] Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from shrinkage. In *FOCS*, pages 111–119, 2012. ECCC:TR12-057. 5, 12, 17, 30, 31

[Juk12] Stasys Jukna. *Boolean Function Complexity - Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012. 9

[KC00] Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *STOC*, pages 73–79, 2000. ECCC:TR99-0045. 3

[LS11] Shachar Lovett and Srikanth Srinivasan. Correlation bounds for poly-size AC^0 circuits with $n^{1-o(1)}$ symmetric gates. In *APPROX/RANDOM*, pages 640–651, 2011. doi:10.1007/978-3-642-22935-0_54. 23

[LVW93] Michael Luby, Boban Velickovic, and Avi Wigderson. Deterministic approximate counting of depth-2 circuits. In *Israel Symposium on Theory of Computing Systems (ISTCS)*, pages 18–24, 1993. doi:10.1109/ISTCS.1993.253488. 23

[Nec66] E.I. Nechiporuk. On a boolean function. *Doklady Akademii Nauk SSSR*, 169(4):765–766, 1966. English translation in Soviet Mathematics Doklady. 3

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994. doi:10.1016/S0022-0000(05)80043-1. 21, 22, 23

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996. doi:10.1006/jcss.1996.0004. 5, 12, 27, 28, 29, 30

[OPS18] Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:158, 2018. ECCC:TR18-158. 3

[OS17] Igor Carboni Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *CCC*, pages 18:1–18:49, 2017. ECCC:TR16-197. 4, 21

[OS18] Igor Carboni Oliveira and Rahul Santhanam. Hardness magnification for natural problems. In *FOCS*, pages 65–76, 2018. ECCC:TR18-139. 3

[Sha49] Claude E. Shannon. The synthesis of two-terminal switching circuits. *Bell Systems Technical Journal*, 28:59–98, 1949. doi:10.1002/j.1538-7305.1949.tb03624.x. 9

[ST18] Rocco A. Servedio and Li-Yang Tan. Luby-Velickovic-Wigderson revisited: Improved correlation bounds and pseudorandom generators for depth-two circuits. In *APPROX/RANDOM*, pages 56:1–56:20, 2018. arXiv:1803.04553. 23, 24

[Tal14] Avishay Tal. Shrinkage of de Morgan formulae by spectral techniques. In *FOCS*, pages 551–560, 2014. ECCC: TR14-048. 3

[Tal17a] Avishay Tal. Formula lower bounds via the quantum method. In *STOC*, pages 1256–1268, 2017. doi:10.1145/3055399.3055472. 3, 24

[Tal17b] Avishay Tal. Tight bounds on the fourier spectrum of AC^0. In *CCC*, pages 15:1–15:31, 2017. ECCC:TR14-174. 18

[Tra84] Boris A. Trakhtenbrot. A survey of russian approaches to perebor (brute-force searches) algorithms. *IEEE Annals of the History of Computing*, 6(4):384–400, 1984. doi:10.1109/MAHC.1984.10036. 3

[TX13] Luca Trevisan and Tongke Xue. A derandomized switching lemma and an improved derandomization of AC^0. In *CCC*, pages 242–247, 2013. ECCC:TR12-116. 5, 18

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

[Vio07] Emanuele Viola. Pseudorandom bits for constant-depth circuits with few arbitrary symmetric gates. *SIAM J. Comput.*, 36(5):1387–1403, 2007. doi:10.1109/CCC.2005.25. 23

[vzGG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2013. 7

[Weg87] Ingo Wegener. *The complexity of Boolean functions*. Wiley-Teubner, 1987. 9

A Circuit complexity of the Nisan-Zuckerman extractor: Proof of Lemma 22

In this section, we will describe the construction of the Nisan-Zuckerman extractor [NZ96], and show that it can be computed by a circuit of almost-linear size.

Lemma 45 (Lemma 22 restated)**.** *There exists an extractor $E\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ that is an $(n/2, \varepsilon)$-extractor with $m = \Omega(n)$ and $d = \mathsf{polylog}(n/\varepsilon)$. Moreover, E can be computed by a circuit of size $n \cdot \mathsf{polylog}(n/\varepsilon)$.*

In proving Lemma 22, we start with some definitions. The extractor works for sources of high min-entropy.

Definition 46 (Dense source)**.** *We say that a distribution over $\{0,1\}^n$ is a* δ-source *if it has min-entropy at least $\delta \cdot n$.*

Definition 47 (Block-wise source)**.** *A distribution $X = (X_1, \ldots, X_s)$ over $\{0,1\}^{\ell_1} \times \cdots \times \{0,1\}^{\ell_s}$ is called a* block-wise δ-source *if, for every $x_1, \ldots, x_i$, $X_i|_{X_1=x_1,\ldots,X_{i-1}=x_{i-1}}$ is a δ-source (i.e., has min-entropy at least $\delta \cdot \ell_i$).*

The extractor will make use of universal hashing, which we define below.

Definition 48 (k-wise independent hashing)**.** *A family of hash functions $\mathcal{H} = \{h\colon \{0,1\}^n \to \{0,1\}^m\}$ is called* k-wise independent *if, for any $x_1, \ldots, x_k \in \{0,1\}^n$, where $x_1, \ldots, x_k$ are distinct, and $y_1, \ldots, y_k \in \{0,1\}^m$, we have*

$$\mathbf{Pr}_{h \sim \mathcal{H}}[h(x_1) = y_1 \wedge \cdots \wedge h(x_k) = y_k] = (1/2^m)^k .$$

$\mathcal{H}$ is also called a universal hash family if it is 2-wise independent.

It is easy to see that any k-wise independent hashing family can be defined using some k-wise independent distribution. As a result, by Lemma 8, we have the following construction of k-wise independent hash families.

Lemma 49. *There exists a k-wise independent hash family $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ such that, given any $h \in \mathcal{H}$, as a kn-bit string, the function h can be computed by a circuit of size $k \cdot \widetilde{O}(\max\{n, m\})$.*

The Nisan-Zuckerman extractor consists of two parts. The first part, block-wise source conversion, takes the source of high min-entropy and converts it into an almost block-wise source by building a list of "blocks." The second part, block-wise source extraction, takes the resulting block-wise source of the previous part and extracts the randomness "block-by-block," using some hash-based extractor. Next, we describe some basic component functions as well as how they are combined to perform the respective task of each part. The main focus here is around the circuit complexity of these procedures and we will not get into details about their correctness. Interested readers are referred to [NZ96, Section 5] for details on the correctness.

In the following, we only work with δ-sources and block-wise δ'-sources where δ and δ' are constants.

Block-wise source converter D. This function has the following parameters:

- n, the size of the original input;
- δ, the quality of the input source;
- $\ell_1 \leq \cdots \leq \ell_s \leq n$, the size of each block; and
- k, the amount of independence used.

We first describe how to build one block using a function that we call B. To build the i-th block, on input $x \in \{0,1\}^n$ and $y_i \in \{0,1\}^{O(k \log n)}$, the function B first divides x into ℓ_i consecutive disjoint sets $A_1, \ldots, A_{\ell_i}$, each of size $m_i = n/\ell_i$. It then uses the $(k \log n)$-bit string y_i to pick, k-wise independently, $j_1, j_2, \ldots, j_{\ell_i}$, where $j_q \in [m_i]$, for each $q \in [\ell_i]$, and outputs the ℓ_i-bit vector

$$x_{j_1} x_{j_2} \cdots x_{j_{\ell_i}}.$$

The block-wise source converter D works as follows.

1. **Input:** $x \in \{0,1\}^n$ and $y_1, \ldots, y_s \in \{0,1\}^{O(k \log n)}$.
2. **Output:** $B(x, y_1), \ldots, B(x, y_s) \in \{0,1\}^{\ell_1} \times \cdots \times \{0,1\}^{\ell_s}$.

Nisan and Zuckerman [NZ96] showed that if the input x is from a δ-source, then for all but at most a $\varepsilon/4$ fraction of the seeds $y_1, \ldots, y_s$, the output of the function D is $(\varepsilon/4)$-close to a block-wise δ'-source, where $\delta' = \Omega(\delta/\log(1/\delta))$.

Claim 50. *The function D can be computed using a circuit of size $s \cdot k \cdot \widetilde{O}(n)$.*

Proof. It is sufficient to show that outputting the i-th block takes a circuit of size $k \cdot \widetilde{O}(n)$. On input $y_i \in \{0,1\}^{k \cdot \log n}$, we can compute, using Lemma 8, $(j_1, j_2, \ldots, j_{\ell_i}) \in [m_i]^{\ell_i}$ with a circuit of size

$$\ell_i \cdot k \cdot \widetilde{O}(\log(m \cdot \ell_i)) = k \cdot \widetilde{O}(n).$$

Then, for each j_q, with $q \in [\ell_i]$, we can compute x_{j_q} using a circuit of size $O(n)$ (by Lemma 14). $\square$

Block-wise source extractor C. This function has $s+1$ parameters:

- δ', the quality of the block source, and
- $\ell_1, \ldots, \ell_s$, the block sizes. Here, $\frac{\ell_{i-1}}{\ell_i} = 1 + \frac{\delta'}{4}$, for all $1 < i \leq s$.

The way the block-wise source extractor C works is described below.

1. **Input:** $x_1 \in \{0,1\}^{\ell_1}, \ldots, x_s \in \{0,1\}^{\ell_s}$ and $y_0 \in \{0,1\}^{2\ell_s}$.
2. We assume for each i a fixed universal family of hash functions
$$\mathcal{H}_i = \left\{ h\colon \{0,1\}^{\ell_i} \to \{0,1\}^{\delta' \ell_i / 2} \right\}_h .$$
Each function in $\mathcal{H}_i$ can be described by $2\ell_i$ bits.
3. $h_s \leftarrow y_0$.
4. For $i \leftarrow s$ down to 1: $h_{i-1} \leftarrow h_i \circ h_i(x_i)$.
5. **Output:** h_0, exculding the bits in h_s. Note that this output is a string in $\{0,1\}^m$.

It was shown in [NZ96] that if $x_1, \ldots, x_s$ are chosen from a block-wise δ'-source and y_0 is uniform, then the output of the function C is $\left(2 \cdot 2^{-\delta' \ell_s / 4}\right)$-close to uniform.

Claim 51. *The function C can be computed using a circuit of size $s \cdot \widetilde{O}(\ell_1)$.*

Proof. Note that given $h_{i-1} \in \{0,1\}^{2\ell_i}$ and $x_i \in \{0,1\}^{\ell_i}$, we can compute $h_i(x_i)$ using a circuit of size $\widetilde{O}(\ell_i)$ (by Lemma 49). Then, to compute h_0, we need to compute h_i for $i = s-1, \ldots, 0$, which takes a circuit of size
$$\sum_{i=1}^{s} \widetilde{O}(\ell_i).$$
The above is at most $s \cdot \widetilde{O}(\ell_1)$, since ℓ_1 is the largest among $\ell_1, \ldots, \ell_s$. ☐

The final extractor E. The parameters are:

- n, the size of the input source;
- δ, the quality of the input source;
- ε, the quality of the output distribution;
- $\delta' = \Theta(\delta / \log(1/\delta))$;
- $\ell_0 = \Theta(\delta^2 n / \log(1/\delta))$; $\ell_i = \ell_{i-1}/(1 + \delta'/4)$ for each $0 < i < s$, with $s = O(\log(n) \log(1/\delta)/\delta)$; therefore, $\ell_s = \log(1/\varepsilon) \log(1/\delta)/\delta$;
- $k = O(\log(n/\varepsilon)$.

The following is the description on the extractor E:

1. **Input:** $x_1 \in \{0,1\}^n$, $y_1, \ldots, y_s \in \{0,1\}^{O(k \log n)}$, $y_0 \in \{0,1\}^{2\ell_s}$.
2. **Output:** $C(D(x, y_1, \ldots, y_s), y_0)$. (Here D and C are used with the parameters specified above.)

It was showed in [NZ96] that if x is from a δ-source and the y's are uniform, then the output of the function E is ε-close to a uniform m-bit string, where $m = \Omega(\delta^2 n / \log(1/\delta))$.

Claim 52. *The function E can be computed using a circuit of size $n \cdot \mathsf{polylog}(n/\varepsilon)$.*

Proof. This follows easily from Claim 50 and Claim 51. ☐

B The IMZ PRG is "almost strongly local"

Here, we show that the IMZ PRG [IMZ12] is "almost strongly local," in the sense that, for most of its seeds, the output of the PRG can be computed by some circuit of size comparable to its seed length.

Lemma 53. *For any $s \geq N$, there exists a PRG $G\colon \{0,1\}^r \to \{0,1\}^N$ that $1/\mathsf{poly}(N)$-fools de Morgan formulas in N variables of size s, where $r = s^{1/3} \cdot 2^{O\left(\log^{2/3} s\right)}$. Moreover, for at least a fraction of $1 - 1/\mathsf{poly}(N)$ of the seeds $z \in \{0,1\}^r$, the function defined as*

$$g_z(j) = G(z)_j$$

can be computed by a circuit of size $s^{1/3} \cdot 2^{O\left(\log^{2/3} s\right)}$.

It is easy to see that such a PRG is sufficient to obtain $\mathtt{MCSP}$ lower bounds using our framework (Theorem 16).

We first need a version of the pseudorandom shrinkage lemma, in which we select and fix the variables both in a pseudorandom manner (note that in Lemma 23 we select the variables pseudorandomly and then fix the variables in a truly-random manner). Such a pseudorandom shrinkage lemma is provided in [IMZ12].

Lemma 54 (Pseudorandom shrinkage lemma, Lemma 4.8 of [IMZ12])**.** *There exists a constant $c_0 > 0$ such that the following hold. For any constant $c > c_0$, any $s \geq N$, $p \geq s^{-1/2}$, and any de Morgan formula F on N variables of size s, there exists a p-regular pseudorandom restriction $\mathcal{D}$ over $\{0,1,*\}^N$ that is samplable using $r = 2^{O(\log^{2/3} s)}$ random bits such that*

$$\mathbf{Pr}_{\rho \sim \mathcal{D}}[L(F_\rho) \geq 2^{3 \cdot c \cdot \log^{2/3} s} \cdot p^2 \cdot s] \leq s^{-c}.$$

Moreover, there exists a circuit of size $2^{O(\log^{2/3} s)}$ such that, given $j \in \{0,1\}^{\log N}$ and a seed $z \in \{0,1\}^r$, the circuit computes the j-th bit of $\mathcal{D}(z)$.

We are now ready to show Lemma 53.

Proof of Lemma 53. The construction is essentially that of [IMZ12]. We use the same parameters as those in the proof of Lemma 18.

The PRG first samples t independent pseudorandom restrictions using Lemma 54. For each of the restrictions, the PRG replaces the $*$ coordinates with the output of some extractor (in fact, it is the output of some limited-independence generator that takes the output of the extractor as a seed). After the $*$ coordinates are replaced in each restriction, the PRG XORs, coordinate-wisely, the t binary strings. More formally, the PRG takes as input a seed

$$(X, Y_1, Y_2, \ldots, Y_t, \gamma_1, \gamma_2, \ldots, \gamma_t) \in \{0,1\}^r,$$

where

- $X \in \{0,1\}^{\aleph}$ is the min-entropy source sample of an extractor,
- $Y_i \in \{0,1\}^{\mathsf{polylog}(N)}$, for each $i \in [t]$, is the seed of an extractor, and
- $\gamma_i \in \{0,1\}^{\ell}$, for each $i \in [t]$, is the seed for sampling a pseudorandom restriction.

Then, the j-th bit of the PRG is the XOR of a sequence of bits $(U_1)_j, (U_2)_j, \ldots, (U_t)_j$, where for each $i \in [t]$ the value of $(U_i)_j$ depends on the value of $(\rho_i)_j$, where ρ_i is a p-regular pseudorandom restriction sampled from Lemma 54 with seed γ_i. Specifically,

$$(U_i)_j = \begin{cases} (\rho_i)_j, & \text{if } (\rho_i)_j \neq *, \text{ and} \\ (Z_i)_j = G_k(E(X, Y_i))_j, & \text{if } (\rho_i)_j = *, \end{cases}$$

where $E\colon \{0,1\}^{\aleph} \times \{0,1\}^d \to \{0,1\}^{\Omega(\aleph)}$ is an $(\aleph/2, \varepsilon)$-extractor, and $G_k\colon \{0,1\}^{r_k} \to \{0,1\}^N$ is a k-independent generator. It was shown in [IMZ12] that the PRG constructed as above ε-fools de Morgan formulas of size s.

Note that, for each $i \in [t]$ and $j \in [N]$, $(\rho_i)_j$ can be computed by a circuit of size $M_1 = 2^{O(\log^{2/3} s)}$ (Lemma 54). Also, using Lemma 22 and Lemma 8, $(Z_i)_j$ can be computed by a circuit of size $M_2 = \widetilde{O}(\aleph) = s^{1/3-o(1)}$.

To compute the j-th bit of the PRG, we need to have the values $(U_1)_j, (U_2)_j, \ldots, (U_t)_j$. It seems that we need to compute both $(\rho_i)_j$ (which is cheap to compute) and $(Z_i)_j$ (which is expensive to compute) for *all* $i \in [t]$, which seems to require size at least $t \cdot M_2 \geq s^{2/3}$. However, we want to compute this with a circuit of size $s^{1/3}$. The key observation here is that we do not need to compute $(Z_i)_j$ for all the i values; we only need to compute $(Z_i)_j$ for those i's such that the j-th coordinate of the i-th pseudorandom restriction is a star (i.e., $(\rho_i)_j = *$). Since the j-th coordinate is a star with probability p, we can expect to see only $p \cdot t \leq O(\log N)$ stars in the sequence $((\rho_i)_j)_{i\in[t]}$. In fact, since the t pseudorandom restrictions are independently sampled, by a standard concentration bound, with very high probability, we only see $\mathsf{polylog}(N)$ stars in the sequence. Then the union bound over the N coordinates yields that, with high probability over the ρ's, we only have $\mathsf{polylog}(N)$ stars in $((\rho_i)_j)_{i\in[t]}$, for all $j \in [N]$. Therefore, for each of these "good" seeds, to compute the j-th bit of the PRG, we can first compute the sequence $((\rho_i)_j)_{i\in[t]}$ (which can be done with a circuit of size $t \cdot M_1$). Then, for each i such that the j-th coordinate of the i-th restriction is a star (there are only $\mathsf{polylog}(N)$-many such i values), we compute $(Z_i)_j$. This can be done by a circuit of size $\mathsf{polylog}(N) \cdot M_2$.

We provide a sketch of how to implement a circuit performing the above task. First, we need to compute the sequence $((\rho_i)_j)_{i\in[t]}$, which can be done by a circuit of size

$$t \cdot 2^{O\left(\log^{2/3} s\right)}.$$

Then, we need to find the i's for which we need to compute $(Z_i)_j$, and select the corresponding Y_i's. This can be done by using divide and conquer and "bins" with fixed $\mathsf{polylog}(N)$ slots to store those indices i. More specifically, we first use one such bin for each i upon $((\rho_i)_j)_{i\in[t]}$. If $(\rho_i)_j$ is a star, we store the value of i as an index to the first (leftmost) slot and leave the other slots "empty." At the next step, we combine two adjacent bins and introduce a new bin that stores the star indices in those two bins; here, the indices are stored in the leftmost slots. After $\log t$ steps, we will have a bin that stores all of the star indices (some of the slots in the bin can be "empty"); the whole procedure can be done by a circuit of size $t \cdot \mathsf{polylog}(N)$. Once we have the indices, we retrieve the corresponding Y_i's (using Lemma 14). We then compute the extractor on each of these Y_i's (with the same min-entropy source sample X), and apply the limited-independence generator on the output of the extractor to get the j-th bit for each of those i's. We also need to make sure that we produce only 0 for those i's that come from the "empty" slots in the bin where the indices are stored. Once we have those bits, we XOR them, and then we XOR the resulting bit with the non-star values in $((\rho_i)_j)_{i\in[t]}$. The XOR of the non-star values can be obtained by XORing the values in $((\rho_i)_j)_{i\in[t]}$ and treating the stars as 0's. $\square$

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il