Algorithmica (2023) 85:2131–2155
https://doi.org/10.1007/s00453-022-01091-y

RESEARCH

Algorithms and Lower Bounds for Comparator Circuits from Shrinkage

Bruno P. Cavalar[1] · **Zhenjian Lu**[2]

Received: 1 August 2022 / Accepted: 29 December 2022 / Published online: 13 January 2023
© The Author(s) 2023

Abstract
In this paper, we initiate the study of *average-case complexity* and *circuit analysis algorithms* for comparator circuits. Departing from previous approaches, we exploit the technique of shrinkage under random restrictions to obtain a variety of new results for this model. Among them, we show

- *Average-case Lower Bounds* For every $k = k(n)$ with $k \geqslant \log n$, there exists a polynomial-time computable function f_k on n bits such that, for every comparator circuit C with at most $n^{1.5}/O\left(k \cdot \sqrt{\log n}\right)$ gates, we have

$$\mathop{\mathbf{Pr}}_{x \in \{0,1\}^n}[C(x) = f_k(x)] \leqslant \frac{1}{2} + \frac{1}{2^{\Omega(k)}}.$$

  This average-case lower bound matches the worst-case lower bound of Gál and Robere by letting $k = O(\log n)$.
- *# SAT Algorithms* There is an algorithm that counts the number of satisfying assignments of a given comparator circuit with at most $n^{1.5}/O\left(k \cdot \sqrt{\log n}\right)$ gates, in time $2^{n-k} \cdot \mathrm{poly}(n)$, for any $k \leqslant n/4$. The running time is non-trivial (i.e., $2^n/n^{\omega(1)}$) when $k = \omega(\log n)$.
- *Pseudorandom Generators and* MCSP *L*ower Bounds There is a pseudorandom generator of seed length $s^{2/3+o(1)}$ that fools comparator circuits with s gates. Also, using this PRG, we obtain an $n^{1.5-o(1)}$ lower bound for MCSP against comparator circuits.

Keywords Comparator circuits · Average-case complexity · Satisfiability algorithms · Pseudorandom generators

✉ Bruno P. Cavalar
Bruno.Pasqualotto-Cavalar@warwick.ac.uk

Zhenjian Lu
zlu1905@gmail.com

[1] Department of Computer Science, University of Warwick, Coventry, UK

[2] Department of Computer Science, University of Oxford, Oxford, UK

1 Introduction

A comparator circuit is a Boolean circuit whose gates are *comparator gates*, each of which maps a pair of inputs (x, y) to $(x \wedge y,\ x \vee y)$, and whose inputs are labelled by a literal (i.e., a variable x_i or its negation $\neg x_i$). A convenient way of representing a comparator circuit is seen in Fig. 1. We draw a set of horizontal lines, each of which is called a *wire* and is labelled by an input literal. The gates are represented as a sequence of vertical arrows, each of which connects some wire to another. The tip of the arrow is the logical disjunction gate ($\vee$), and the rear of the arrow is the logical conjunction gate ($\wedge$). One of the wires is selected to represent the Boolean value of the computation. The *size* of the circuit is the number of gates in the circuit.

Comparator circuits can be viewed as a restricted type of circuit in which the gates have bounded fan-out. It is easy to see that comparator circuits can efficiently simulate Boolean formulas over $\{\wedge, \vee, \neg\}$ with no overhead.[1] Moreover, it is also known that polynomial-size comparator circuits can even simulate nondeterministic branching programs [2] with a polynomial overhead only. On the other hand, comparator circuits appear to be much stronger than formulas,[2] as it is conjectured that polynomial-size comparator circuits are incomparable to NC [2]. Evidence for this conjecture is that polynomial-size comparator circuits can compute problems whose known algorithms are inherently sequential, such as stable marriage and lexicographically first maximal matching [2], and there is an oracle separation between NC and polynomial-size comparator circuits [3]. Moreover, Robere, Pitassi, Rossman and Cook [4] showed that there exists a Boolean function in mNC^2 not computed by polynomial-size *monotone* comparator circuits.[3] For these reasons, comparator circuits are likely to be incomparable to NC, and polynomial-size span programs, which are contained in NC^2, are not expected to be stronger than polynomial-size comparator circuits.

Despite the importance of comparator circuits, we don't know much about them. Though it is easy to see that Parity can be computed by comparator circuits with $O(n)$ wires and gates (see Fig. 1), the best known comparator circuit for Majority uses $O(n)$ wires and $O(n \log n)$ gates [5]. We don't know if there is a linear-size comparator circuit for Majority,[4] whereas, for the weaker model of nondeterministic branching programs, superlinear lower bounds are known [6]. Structural questions about comparator circuits have also received some attention in recent years [7, 8].

The first superlinear *worst-case lower bound* for comparator circuits was recently obtained by Gál and Robere [9], by an adaptation of Nečiporuk's argument [10]. Their proof yields a lower bound of $\Omega\big((n/\log n)^{1.5}\big)$ to comparator circuits computing a function of n bits. For *monotone* comparator circuits, exponential lower bounds are known [4].

In this paper, we exploit structural properties of small-size comparator circuits in order to prove the first *average-case lower bounds* and design the first *circuit analysis*

[1] As a comparison, note that there are linear-size comparator circuits for Parity (see Fig. 1), whereas any Boolean formula computing Parity has size $\Omega(n^2)$ [1].

[2] Recall that the class of polynomial-size formulas is exactly NC^1.

[3] Comparator circuits are monotone if they don't have negated literals.

[4] As opposed to a sorting network, note that a comparator circuit can use multiple copies of the same input literal.

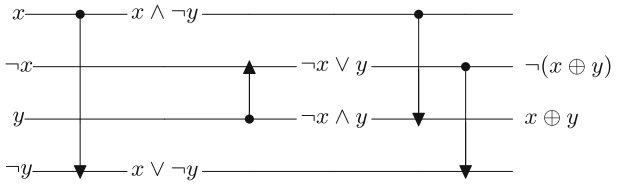

Fig. 1 A comparator circuit with 2 inputs, 4 wires and 4 gates. The third wire computes the parity of the two bits

algorithms for small comparator circuits. Developing circuit analysis algorithms is a crucial step for understanding a given circuit class [11, 12], and are often obtained only after lower bounds have been proven for the class.[5] Many well-studied circuit classes have been investigated from this perspective, such as AC^0 circuits [14], De Morgan formulas [15], branching programs [16], ACC circuits [13], and many others (see also [17–19]). Our paper commences an investigation of this kind for comparator circuits.

1.1 Results

Average-case lower bounds Our work starts with the first average-case lower bound against comparator circuits.

Theorem 1 (Average-case Lower Bound) *There exist constants $c, d \geqslant 1$ such that the following holds. For any $k \geqslant c \cdot \log n$, there is a polynomial-time computable function f_k such that, for every comparator circuit C with at most*

$$\frac{n^{1.5}}{d \cdot k \cdot \sqrt{\log n}}$$

gates, we have

$$\Pr_{x \in \{0,1\}^n}[f_k(x) = C(x)] \leqslant \frac{1}{2} + \frac{1}{2^{\Omega(k)}}.$$

An important feature of the lower bound in Theorem 1 is that it matches the $\Omega\big((n/\log n)^{1.5}\big)$ worst-case lower bound of [9], in the sense that we can recover their result (up to a multiplicative constant) by setting $k = O(\log n)$.

Using ideas from the proof of the above average-case lower bound, we also show average-case lower bounds against various models that tightly match their state-of-the-art worst-case lower bounds, such as general formulas, (deterministic-, nondeterministic-, parity-)branching programs and span programs (see Sect. 4). Note that strong average-case lower bounds against $n^{2-o(1)}$-size general formulas and deterministic branching programs were previously known [17, 20] but they did not match

[5] One exception is ACC circuits, for which satisfiability algorithms are known [13], and the only exponential lower bound known for ACC is a consequence of this algorithm. However, the function used in the lower bound is not in NP.

the worst-case lower bounds, whereas tight average-case lower bounds for *De Morgan* formulas were proved by [21].

SAT algorithms The design of algorithms for interesting *circuit analysis problems* is a growing line of research in circuit complexity [12]. These are problems that take circuits as inputs. A famous example of such a circuit analysis problem is the *satisfiability* problem (SAT), which asks to determine whether a given circuit has a satisfying assignment. Note that the satisfiability problem for polynomial-size general circuits is NP-complete, so it is not believed to have a polynomial-time (or subexponential-time) algorithm. However, one can still ask whether we can obtain *non-trivial* SAT algorithms running faster than exhaustive search, say in time $2^n/n^{\omega(1)}$ where n is the number of variables of the input circuit, even for restricted circuit classes. While designing non-trivial SAT algorithms is an interesting problem by itself, it turns out that this task is also tightly connected to proving lower bounds. In particular, recent works of Williams [13, 22] have shown that such a non-trivial satisfiability algorithm for a given class of circuits can often be used to prove non-trivial circuit lower bounds against that same circuit class.

Here, we show an algorithm with non-trivial running time that counts the number of satisfying assignments of a given comparator circuit.

Theorem 2 (#SAT Algorithms) *There is a constant $d > 1$ and a deterministic algorithm such that, for every $k \leqslant n/4$, given a comparator circuit on n variables with at most*

$$\frac{n^{1.5}}{d \cdot k \cdot \sqrt{\log n}}$$

gates, the algorithm outputs the number of satisfying assignments of C in time

$$2^{n-k} \cdot \mathrm{poly}(n).$$

Note that the running time in Theorem 2 is non-trivial for size up to $o(n/\log n)^{1.5}$, in which case $k = \omega(\log n)$ and the running time becomes $2^n/n^{\omega(1)}$.

Pseudorandom generators and MCSP *lower bounds.* Another important circuit analysis problem is *derandomization*, which, roughly speaking, asks to decide whether a given circuit accepts or rejects a large fraction of its inputs. A standard approach to solve this problem is to construct a pseudorandom generator (PRG). A PRG against a class $\mathcal{C}$ of circuits is an efficient and deterministic procedure G mapping short binary strings (seeds) to longer binary strings, with the property that G's output (over uniformly random seeds) "looks random" to every circuit in $\mathcal{C}$. More precisely, we say that a generator $G\colon \{0,1\}^r \to \{0,1\}^n$ *ε-fools* a class $\mathcal{C}$ of circuits if, for every $C\colon \{0,1\}^n \to \{0,1\}$ from $\mathcal{C}$, we have

$$\left| \mathop{\mathbf{Pr}}_{z\in\{0,1\}^r}[C(G(z)) = 1] - \mathop{\mathbf{Pr}}_{x\in\{0,1\}^n}[C(x) = 1] \right| \leqslant \varepsilon,$$

In constructing PRGs, we aim to minimize the parameter r, which is called the seed length.

We show a PRG against comparator circuits of size s with seed length $s^{2/3+o(1)}$.

Theorem 3 (Pseudorandom Generators) *For every $n \in \mathbb{N}$, $s = n^{\Omega(1)}$, and $\varepsilon \geqslant 1/\operatorname{poly}(n)$, there is a pseudorandom generator $G\colon \{0,1\}^r \to \{0,1\}^n$, with seed length*

$$r = s^{2/3+o(1)},$$

that ε-fools comparator circuits on n variables with s gates.

Note that the seed length of the PRG in Theorem 3 is non-trivial (i.e., $o(n)$) for comparator circuits of size $n^{1.5-o(1)}$.

The PRG above has an application in obtaining lower bounds for the *minimum circuit size problem* (MCSP) against comparator circuits. The MCSP problem asks if a given truth table[6] represents a function that can be computed by some small-size circuit. Understanding the exact complexity of MCSP is a fundamental problem in complexity theory. Motivated by a recent line of research called *hardness magnification* [23–26], which states that a weak circuit lower bound for certain variants of MCSP implies breakthrough results in circuit complexity, researchers have been interested in showing lower bounds for MCSP against restricted classes of circuits. For many restricted circuit classes such as constant-depth circuits, formulas and branching programs, the lower bounds that have been proved for MCSP essentially match the best known lower bounds that we have for any explicit functions [19, 27, 28]. Here we obtain MCSP lower bounds against comparator circuits that nearly match the worst-case lower bounds.

Theorem 4 (MCSP Lower Bounds) *Let* MCSP$[n^\alpha]$ *denote the problem of deciding whether a given n-bit truth table represents a function that can be computed by some general circuit of size at most n^α. For any $\varepsilon > 0$ and any $0 < \alpha \leqslant 1-\varepsilon$, the* MCSP$[n^\alpha]$ *problem does not have comparator circuits with $n^{1+\alpha/2-\varepsilon}$ gates.*

Previously, non-trivial comparator circuit lower bounds were known only for functions satisfying Nečiporuk's criterion [9, 10], such as *Element Distinctness* and *Indirect Storage Access*. Theorem 4 provides yet another natural computational problem which is hard for bounded-size comparator circuits. We remark that the MCSP problem is expected to require much larger circuits than the lower bound of Theorem 4 provides; however, the lack of combinatorial, algebraic or analytic structure in the MCSP function means that proving lower bounds for it is usually hard.

Finally, we also observe that the framework developed in [18] can be used to obtain a *non-trivial* (distribution-independent) PAC learning algorithm for comparator circuits of size $n^{1.5-o(1)}$, that uses membership queries (see Sect. 7).

1.2 Techniques

Random restrictions have been very fruitful in the study of weaker circuit classes, such as AC^0 circuits [14, 29], De Morgan formulas [20] and branching programs [16], both for the proof of lower bounds and the construction of algorithms [17]. However, as

[6] A truth table is a bit-string that stores the output values of a Boolean function for all possible inputs.

observed by Gál and Robere [9], there are technical challenges when trying to apply this approach to comparator circuits. In this work, we successfully apply the method of random restrictions to comparator circuits for the first time.

Average-case lower bounds At a high level, the proof of our average-case lower bound is based on the approach developed in [17, 20], which can be used to obtain average-case-lower bounds against circuits that admit a property called "shrinkage with high probability under random restrictions". Roughly speaking, this property says that, if we randomly fix the values of some variables in the circuit except for a $0 < p < 1$ fraction of them, then its size shrinks by a factor of p^{Γ} for some $\Gamma > 0$, with very high probability. This method has been used to obtain strong average-case lower bounds against $n^{2.5-o(1)}$-size De Morgan formulas [17, 20] (later improved to $n^{3-o(1)}$ by [21]) and $n^{2-o(1)}$-size general formulas and deterministic branching programs.

An obvious issue of applying this approach to comparator circuits is that we don't know how to shrink the size (i.e., number of gates) of a comparator circuit using random restrictions, as when we fix the value of a (non-trivial[7]) wire, we may only be able to remove one gate in the worst scenario (i.e., the gate that is directly connected to that wire). The idea is that instead of shrinking the *number of gates*, we will try to shrink the *number of wires*. The reason why this can be useful is that *one can effectively bound the number of gates of a comparator circuit by its number of wires*; this is a structural result of comparator circuits proved by Gál and Robere [9] and was the key ingredient in proving their worst-case lower bound. More precisely, they showed that any comparator circuit that has at most ℓ wires needs no more than ℓ^2 gates (see Lemma 8). Now following [17, 20], one can show that under some certain type of random restriction that leaves a $p := k/n$ fraction of the variables unfixed, for any large enough k, the number of wires of a comparator circuit will shrink (with very high probability) by roughly a factor of p, and hence its number of gates is bounded by $(p \cdot \ell)^2$. By letting $\ell = o\big(n^{1.5}/\big(k \cdot \sqrt{\log n}\big)\big)$, this size is less than $o(n/\log n)$ and from there one can show that the original circuit cannot compute some hard function on more than $1/2 + 1/2^{k^{\Omega(1)}}$ fraction of the inputs.

While the above gives an average-case lower bound, it does not match the worst-case one, because we need to set $k \geqslant \log^c n$ for some (unspecified) constant $c > 1$, which is controlled by the type of random restrictions and the extractor used in the construction of the hard function in both [17, 20]. This means we can only achieve a lower bound that is at best $n^{1.5}/(\log n)^{c+.5}$ (even for worst-case hardness). In order to be able to set $k = O(\log n)$, one way is to use a more sophisticated (so called non-explicit bit-fixing) extractor shown in [21], which will allow us to set $k \in \big[O(\log n)\,,\,\Omega\big(n^{1/3}\big)\big]$ (with hardness $1/2 + 1/2^{\Omega(k)}$). Here we refine and simplify this approach in the case of comparator circuits by using a more structural (block-wise) random restriction that shrinks the number of wires with probability one. Such a random restriction, when combined with a simple extractor, allows us to set $k \in \big[O(\log n)\,,\,\Omega(n)\big]$.

 # SAT algorithms Based on the above analysis in showing average-case lower bounds, one can try to design a SAT algorithm for comparator circuits in a way that is similar to that of [17], which combines "shrinkage under restrictions" with a memorization

[7] We say that a wire is non-trivial if it is connected to some gate.

technique. Suppose we have a comparator circuit C with $s := o\big(n^{1.5}/(k \cdot \sqrt{\log n})\big)$ gates and and $\ell \leqslant s$ non-trivial wires. By partitioning the variables into n/k equal-size blocks, we can show that there is some block S_i such that after fixing the variables outside of this block, the number of wires in the restricted circuit is at most $\ell_0 := \ell/(n/k) \leqslant o\big(\sqrt{n/\log n}\big)$. Again by the structural property of comparator circuits (Lemma 8), this restricted circuit, which is on k variables, has an equivalent circuit with $o(n/\log n)$ gates. Then to count the number of satisfying assignments for the original circuit, we can first memorize the numbers of satisfying assignments for all circuits with at most with $o(n/\log n)$ gates. There are $2^{o(n)}$ of them and hence we can compute in time $2^k \cdot 2^{o(n)}$ a table that stores those numbers. We then enumerate all possible 2^{n-k} restrictions $\rho \in \{0,1\}^{[n]\setminus S_i}$ and for each ρ we look up the number of satisfying assignments of the restricted circuit $C\restriction_\rho$ from the pre-computed table. Summing these numbers over all the ρ's gives the number of satisfying assignments of C.

However, there is a subtle issue in the above argument: although we know that a restricted circuit has an equivalent simple circuit with $o(n/\log n)$ gates, we do not know which simple circuit it is equal to. Note that when we fix the value of a (non-trivial) wire, we may only be able to remove one gate, so the number of gates left in the restricted circuit is possibly $s - (\ell - \ell_0)$, which can be much larger than $n/\log n$, and it is not clear how we can further simplify such a circuit *efficiently*. To overcome this issue, we explore structural properties of comparator circuits to show how to construct a more sophisticated data structure that not only can tell us the number of satisfying assignments of a circuit with $o(n/\log n)$ gates but also allows us to *efficiently* simplify each restricted circuit to an equivalent circuit with at most this many gates.

Pseudorandom generators and MCSP *lower bounds* Our PRG against comparator circuits builds upon the paradigm of [16], which was used to construct PRGs against circuits that admit "shrinkage under *pseudorandom* restrictions". As in the proof of our average-case-lower bound, in order to apply this paradigm, we will shrink the number of wires instead of the number of gates. Following [16], we prove a pseudorandom shrinkage lemma for comparator circuits, which can then be used to obtain a PRG of seed length $s^{2/3+o(1)}$, where s is the size of a comparator circuit.

As observed in [28], one can modify the construction of the PRG in [16] to make it "locally explicit". This means that, for every seed, the output of the PRG, when viewed as a truth table of a function, has circuit complexity that is about the same as the seed length. Such a "local" PRG immediately implies that MCSP cannot be computed by comparator circuits of size $n^{1.5-o(1)}$, when the size parameter of MCSP is nearly-maximum (i.e., $n/O(\log n)$).[8] Furthermore, we show a better trade-off between the size parameters of MCSP and the lower bound size of the comparator circuits, as in Theorem 4. This is similar to what was done by [30] in the case of MCSP lower bounds against De Morgan formulas.

[8] Note that MCSP takes two input parameters: a truth table and a size parameter θ, and asks whether the given truth table has circuit complexity at most θ.

1.3 Directions and Open Problems

We now state some further directions and open problems for which our work may be a starting point, or that are connected to our results.

Algorithms and lower bounds for larger comparator circuits Our lower bounds and circuit analysis algorithms only work for comparator circuits of size up to $n^{1.5-o(1)}$. Can we improve this? Specifically, can we show a lower bound of $n^{1.51}$ for comparator circuits computing a function of n bits, and design algorithms for comparator circuits of the same size? In this paper, we used the random restriction method to analyse comparator circuits by shrinking the number of wires and using a structural result of [9] that relates the number of gates to the number of wires. Can we analyse the effect of random restrictions on the gates *directly*, and show a shrinkage lemma for comparator circuits on the number of gates, with a shrinkage exponent $\Gamma > 1/2$? Such a lemma would imply a lower bound that is better than $n^{1.5}$, and would allow us to design algorithms for comparator circuits larger than $n^{1.5}$.

Hardness magnification near the state-of-the-art Recent work on *hardness magnification* [23–26] has shown that marginally improving the state-of-art worst-case lower bounds in a variety of circuit models would imply major breakthroughs in complexity theory. Although we don't prove this here, it is possible to show hardness magnification results for comparator circuits of size $n^{2+o(1)}$ by a simple adaptation of their arguments. Unfortunately, this does not match the best lower bounds we have for comparator circuits, which are around $n^{1.5-o(1)}$ as we have seen. Can we show a hardness magnification phenomenom nearly matching the state-of-art lower bounds for comparator circuits?

Extensions and restrictions of comparator circuits Recent work of Komarath, Sarma and Sunil [8] has provided characterisations of various complexity classes, such as L, P and NP, by means of extensions or restrictions of comparator circuits. Can our results and techniques applied to comparator circuits be extended to those variations of comparator circuits? Can this extension shed any light into the classes characterised by [8]?

2 Preliminaries

2.1 Definitions and Notations

For $n \in \mathbb{N}$, we denote $\{1, \ldots, n\}$ by $[n]$. For a string x, we denote by $\mathrm{K}(x)$ the *Kolmogorov complexity* of x, which is defined as the minimum length of a Turing machine that prints x as output.

Restrictions A *restriction* for an n-variate Boolean function f, denoted by $\rho \in \{0, 1, *\}^n$, specifies a way of fixing the values of some subset of variables for f. That is, if $\rho(i)$ is $*$, we leave the ith variable unrestricted and otherwise fix its value to be $\rho(i) \in \{0, 1\}$. We denote by $f\restriction_\rho \colon \{0,1\}^{\rho^{-1}(*)} \to \{0, 1\}$ the restricted function

after the variables are restricted according to ρ, where $\rho^{-1}(*)$ is the set of unrestricted variables.

Comparator circuits We define comparator circuits as a set of *wires* labelled by an input literal (a variable x_i or its negation $\neg x_i$), a sequence of *gates*, which are *ordered* pairs of wires, and a designated *output wire*. In other words, each gate is a pair of wires (w_i, w_j), denoting that the wire w_i receives the logical conjunction ($\wedge$) of the wires, and w_j receives the logical disjunction ($\vee$). On a given input a, a comparator circuit computes as follows: each wire labelled with a literal x_i is initialised with a_i, and we update the value of the wires by following the sequence of gates; the output wire contains the result of the computation. A wire is called *non-trivial* if there is a gate connected to this wire. Note that, if a comparator circuit has ℓ non-trivial wires and s gates, then $\ell \leqslant s$. This means that lower bounds on the number of wires also imply lower bounds on the number of gates.

2.2 Structural Properties of Comparator Circuits

For a gate g in a comparator circuit and an input $x \in \{0,1\}^n$, we denote by $u_g(x)$ (resp. $v_g(x)$) the first (resp. second) in-value to the gate g when given x as input to the circuit.

Definition 5 (*Useless Gates*) We say that a gate g in a comparator circuit is *useless* if either one of the following is true:

1. for every input x, $(u_g(x), v_g(x)) \in \{(0,1), (0,0), (1,1)\}$.
2. for every input x, $(u_g(x), v_g(x)) \in \{(1,0), (0,0), (1,1)\}$.

We say that a useless gate is of *TYPE-1* (resp. *TYPE-2*) if it is the first (resp. second) case. Also, a gate is called *useful* if it is not useless.

The following proposition allows us to remove useless gates from a comparator circuit.

Proposition 6 [9, Proof of Proposition 3.2] *Let C be a comparator circuit whose gates are $g_1, g_2, \ldots, g_s$ (where g_s is the output gate) and let $g_i = (\alpha, \beta)$ be any useless gate in C.*

- *Suppose g_i is of TYPE-1. Then the circuit C' obtained from C by removing the gate g_i computes the same function as that of C.*
- *Suppose g_i is of TYPE-2. Let C' be the circuit whose gates are $g_1, g_2, \ldots, g_{i-1}, g'_{i+1}$ $\ldots, g'_s$, where for $j = i+1, \ldots, s$, g'_j is obtained from g_j by replacing α with β (if g_j contains α) and at the same time replacing β with α (if g_j contains β). Then C' computes the same function as that of C.*

Proof On the one hand, if g is a TYPE-1 useless gate, then for every input to the circuit, the out-values of g are the same as its in-values, so removing g does not affect the function computed by the original circuit. On the other hand, if g is of TYPE-2, then the in-values feeding to g will get swapped after g is applied. This has the same effect as removing g and "re-wiring" the gates after g so that a gate connecting one of the wires of g gets switched to connect the other wire of g, as described in the second item of the proposition. □

We need the following powerful structural result for comparator circuits from [9].

Theorem 7 [9, Theorem 1.2] *If C be is a comparator circuit with ℓ wires and s gates such that every gate in C is useful, then $s \leqslant \ell \cdot (\ell - 1)/2$.*

Proposition 6 and Theorem 7 together give the following lemma.

Lemma 8 *Every comparator circuit with $\ell > 0$ wires has an equivalent comparator circuit with ℓ wires and with at most $\ell \cdot (\ell - 1)/2$ gates.*

3 Average-Case Lower Bounds

In this section, we prove our average-case lower bound against comparator circuits. We first describe the hard function.

3.1 The Hard Function

List-decodable codes Recall that a (ζ, L)-list-decodable binary code is a function $\mathrm{Enc}\colon \{0,1\}^n \to \{0,1\}^m$ that maps n-bit messages to m-bit codewords so that, for each $y \in \{0,1\}^m$, there are at most L codewords in the range of Enc that have relative hamming distance at most ζ from y. We will use the following list-decodable code.

Theorem 9 (See e.g., [17, Proof of Theorem 6.4]) *There is a constant $c > 0$ such that for any given $k = k(n) > c \cdot \log n$, there exists a binary code* Enc *mapping n-bit message to a codeword of length 2^k, such that* Enc *is (ζ, L)-list-decodable for $\zeta = 1/2 - O\left(n/2^{k/2}\right)$ and $L \leqslant O\left(2^{k/2}/n\right)$. Furthermore, there is a polynomial-time algorithm for computing the ith bit of* $\mathrm{Enc}(x)$*, for any inputs $x \in \{0,1\}^n$ and $i \in \left[2^k\right]$.*

Definition 10 (*Generalized Andreev's Function*) Let k be a positive integer. Define $A_k\colon \{0,1\}^{n+n} \to \{0,1\}$ as follow:

$$A_k(x_1, \ldots, x_n, y_1, \ldots, y_n) := \mathrm{Enc}(x_1, \ldots, x_n)_{\alpha(y_1, \ldots, y_n)},$$

where Enc is the code from Theorem 9 that maps n bits to 2^k bits, and $\alpha\colon \{0,1\}^n \to \{0,1\}^k$ is defined as

$$\alpha(y_1, \ldots, y_n) := \left(\bigoplus_{i=1}^{n/k} y_i, \bigoplus_{i=n/k+1}^{2n/k} y_i, \ldots, \bigoplus_{i=(k-1)n/k+1}^{n} y_i \right).$$

That is, the function α partitions y evenly into k consecutive blocks and outputs the parities of the variables in each block.

Note that the function A_k defined above is polynomial-time computable since we can compute $\alpha(y)$ and $\mathrm{Enc}(x)_i$ for any given i in $\mathrm{poly}(n)$ time.

3.2 Proof of the Average-Case Lower Bound

We will show a lower bound on the number of wires, which automatically implies a lower bound on the number of gates.

Theorem 11 *There exist constants $c, d \geqslant 1$ such that the following holds. For any $k \geqslant c \cdot \log n$, there is a polynomial-time computable function f_k such that, for every comparator circuit C whose number of wires is*

$$\frac{n^{1.5}}{d \cdot k \cdot \sqrt{\log n}},$$

we have

$$\Pr_{x \in \{0,1\}^n}[f_k(x) = C(x)] \leqslant \frac{1}{2} + \frac{1}{2^{\Omega(k)}}.$$

Proof Let A_k be the generalized Andreev's function on $2n$ variables. Let C be a comparator circuit on $2n$ variables with $\ell \leqslant n^{1.5}/\left(d \cdot k \cdot \sqrt{\log n}\right)$ wires, where $d \geqslant 1$ is a sufficiently large constant. To avoid some technicalities due to divisibility that can be overcome easily, we assume that n is divisible by k.

We need to upper bound the following probability.

$$\begin{aligned}
&\Pr_{x,y \in \{0,1\}^n \times \{0,1\}^n}[A_k(x, y) = C(x, y)] \\
&\quad \leqslant \Pr_{x,y}[A_k(x, y) = C(x, y) \mid \mathrm{K}(x) \geqslant n/2] + \Pr_{x}[\mathrm{K}(x) < n/2] \\
&\quad \leqslant \Pr_{x,y}[A_k(x, y) = C(x, y) \mid \mathrm{K}(x) \geqslant n/2] + \frac{1}{2^{n/2}}.
\end{aligned}$$

Let x be any fixed n-bit string with Kolmogorov complexity at least $n/2$. Let $A' \colon \{0,1\}^n \to \{0,1\}$ be

$$A'(y) := A_k(x, y),$$

and let C' be a comparator circuit on n variables with at most ℓ wires defined as

$$C'(y) := C(x, y).$$

We will show that

$$\Pr_{y \in \{0,1\}^n}\left[A'(y) = C'(y)\right] \leqslant \frac{1}{2} + \frac{n}{2^{k/4}}.$$

First of all, let us divide the n variables of C' into n/k parts, each of which contains k variables, as follows. We first partition the n variables evenly into k *consecutive*

blocks, denoted as $B_1, B_2, \ldots, B_k$. Then we define the ith part S_i, where $i \in [n/k]$, to be the union of the ith variables in each of $B_1, B_2, \ldots, B_k$. That is

$$S_i := \bigcup_{j\in[k]} \left\{ y : y \text{ is the } i\text{th variables of } B_j \right\}.$$

Now we count the number of wires that are labelled by the variables in each S_i and let

$$w_i := |\{u : u \text{ is a wire labelled by some } x \in S_i \text{ (or its negation)}\}|.$$

We have

$$\sum_{i\in[n/k]} w_i = \ell,$$

which implies that there is a particular $i \in [n/k]$ such that

$$w_i \leqslant \frac{\ell}{n/k} \leqslant \frac{1}{d} \cdot \sqrt{\frac{n}{\log n}} =: \ell_0.$$

Next, we will consider restrictions that fix the values of the variables outside S_i. Note that if we fix the value of a variable x_i in a comparator circuit, then we can obtain a restricted circuit so that all the wires that are labelled by either x_i or $\neg x_i$ are eliminated, after some appropriate updates on the gates in the circuit. This is not an obvious fact. One way to see this is that once we fix the value of a wire, the gate that directly connects this wire becomes *useless* in the sense of Definition 5 so it can be removed after some appropriate "re-wirings" of the gates in the circuit as described in Proposition 6. Then we can keep doing this until no gate is connected to that wire, in which case the wire can be removed from the circuit.

Now we have

$$\mathop{\mathbf{Pr}}_{y\in\{0,1\}^n}\left[A'(y) = C'(y)\right] = \mathop{\mathbf{Pr}}_{\rho\in\{0,1\}^{[n]\setminus S_i}, z\in\{0,1\}^k}\left[A'\restriction_\rho (z) = C'\restriction_\rho (z)\right].$$

It suffices to upper bound

$$\mathop{\mathbf{Pr}}_{z\in\{0,1\}^k}\left[A'\restriction_\rho (z) = C'\restriction_\rho (z)\right],$$

for every $\rho \in \{0,1\}^{[n]\setminus S_i}$. For the sake of contradiction, suppose for some ρ, we have

$$\frac{1}{2} + \frac{n}{2^{k/4}} < \mathop{\mathbf{Pr}}_{z\in\{0,1\}^k}\left[A'\restriction_\rho (z) = C'\restriction_\rho (z)\right] = \mathop{\mathbf{Pr}}_{z\in\{0,1\}^k}\left[\mathrm{Enc}(x)_\alpha = C'\restriction_\rho (z)\right], \quad (1)$$

where $\alpha \in \{0,1\}^k$ is

$$\alpha_j := \mathsf{Parity}\left(\rho\restriction_{B_j \setminus S_i}\right) \oplus z_j,$$

and $\rho\restriction_{B_j \setminus S_i}$ denotes the partial assignment given by ρ but restricted to only variables in the set $B_j \setminus S_i$. Note that α is uniformly distributed for uniformly random z. Therefore, if we have the values of $\mathsf{Parity}\left(\rho\restriction_{B_j \setminus S_i}\right)$ for each $j \in [k]$ (k bits in total), and if we know the restricted circuit $C'\restriction_\rho$, then we can compute the codeword $\mathrm{Enc}(x)$ correctly on at least $1/2 + n/2^{k/4}$ positions, by evaluating $C'\restriction_\rho(z)$ for every $z \in \{0,1\}^k$. As a result, we can list-decode $\mathrm{Enc}(x)$, and, using additional $k/2$ bits (to specify the index of x in the list), we can recover x exactly. Finally, note that the number of wires in $C'\restriction_\rho$ is at most ℓ_0. Therefore, by Lemma 8, such a circuit can be described using a string of length at most

$$\begin{aligned} O\left(\ell_0 \cdot \log(n) + \ell_0^2 \cdot \log(\ell_0)\right) &\leqslant O\left(\ell_0^2 \cdot \log n\right) \\ &= O\left(\frac{n}{d^2 \cdot \log n} \cdot \log n\right) \\ &\leqslant n/4, \end{aligned}$$

where the last inequality holds when d is sufficiently large. Therefore, we can recover x using less than

$$n/4 + k + k/2 + O(\log n) < n/2$$

bits. Here we assume $k \leqslant n/8$ since otherwise the theorem can be shown trivially. This contradicts the fact that the Kolmogorov complexity of x is at least $n/2$. □

4 Tight Average-Case Lower Bounds from a Nečiporuk-Type Property

Here, we describe a generalization of the average-case lower bound in Sect. 3 to circuit classes whose worst-case lower bounds can be proved via Nečiporuk's method.

Theorem 12 *There is a constant $c > 1$ such that the following holds. Let $\mathcal{C}$ be a class of Boolean circuits that is closed under restrictions. Suppose that, for any $k \in [c \cdot \log n,\ n/6]$, there exists a partition of the n variables into $m := n/k$ equal-sized blocks $S_1, S_2, \ldots, S_m$ and a collection of k-input-bit functions $\mathcal{H}$ such that*

1. *$|\mathcal{H}| \leqslant 2^{n/2}$, and*
2. *for every $C \in \mathcal{C}_n$ of size $s(n,k)$, there exists some block S_i such that $\left\{C\restriction_\rho\right\}_{\rho \in \{0,1\}^{[n]\setminus S_i}} \subseteq \mathcal{H}$.*

Then for any $k \in [c \cdot \log n,\ n/6]$, there exists a polynomial-time computable function f_k which satisfies

$$\Pr_{x \in \{0,1\}^n}[C(x) = f_k(x)] \leqslant \frac{1}{2} + \frac{1}{2^{\Omega(k)}},$$

for every $C \in \mathcal{C}_n$ of size $s(n/2, k)$.

Remark In the original Nečiporuk's argument for getting worst-case lower bounds, it is only required that, for every $C \in \mathcal{C}$, there is some block such that the number of distinct functions, after fixing the variables outside of the block, is at most $2^{n/2}$, and *this set of functions can be different for different C*. For Theorem 12, we need something stronger which says that it is the *same set of $2^{n/2}$ functions for every C*. We remark that, though the weaker condition is sufficient for *worst-case* lower bounds, all applications of Nečiporuk's method known to us also prove the stronger condition, thus yielding *average-case* lower bounds by Theorem 12.

Theorem 12 requires a slightly different argument than that of Theorem 11. Its proof is presented in "Appendix A".

By combining Theorem 12 with known structural properties for various models (see e.g., [31]), we get that for the class of circuits $\mathcal{C}$ of size s, where

- $\mathcal{C}$ is the class of general formulas, and $s = n^2/O(k)$, or
- $\mathcal{C}$ is the class of deterministic branching programs or switching networks, and $s = n^2/O(k \cdot \log n)$, or
- $\mathcal{C}$ is the class of nondeterministic branching programs, parity branching programs, or span programs, and $s = n^{1.5}/O(k)$,

there exists a function f_k such that $\mathop{\mathbf{Pr}}_{x \in \{0,1\}^n}[C(x) = f_k(x)] \leqslant 1/2 + 1/2^{\Omega(k)}$ for every $C \in \mathcal{C}$, which matches the state-of-the-art worst-case lower bounds (up to a multiplicative constant) by letting $k = O(\log n)$.

5 #SAT Algorithms

In this section, we present our #SAT algorithm for comparator circuits. As mentioned briefly in Sect. 1.2, we will need a preprocessed data structure that enables us to efficiently convert a circuit with small number of wires but large number of gates to an equivalent circuit (with the same number of wires) whose number of gates is at most quadratic in the number wires.

5.1 Memorization and Simplification of Comparator Circuits

Lemma 13 *Let $n, \ell \geqslant 1$ be integers. For any fixed labelling of ℓ wires on n variables, there is a data structure DS such that*

- *DS can be constructed in time $2^n \cdot \ell^{O(\ell^2)}$.*
- *Given access to DS and given any comparator circuit C with ℓ wires (whose labelling is consistent with the one used for DS) and s gates, we can output in time $\mathrm{poly}(s, \ell)$ the number of satisfying assignments of C. Moreover, we obtain a comparator circuit with ℓ wires and at most $\ell \cdot (\ell - 1)/2$ gates that is equivalent to C.*

Proof We know that every comparator circuit with ℓ wires has an equivalent circuit with $\ell \cdot (\ell - 1)/2$ gates (Lemma 8). Therefore, we can try to memorize the number of satisfying assignments for each of these circuits (by brute-force). Then for a given circuit C with ℓ wires and s gates where $s \gg \text{poly}(\ell)$, we need to simplify C to be a circuit with $\ell \cdot (\ell - 1)/2$ gates so that we can look up its number of satisfying assignments, which was already computed. However, it is not clear how we can *efficiently* simplify such a comparator circuit.

The idea here is to remove the useless gates one by one (from left to right). To do this, firstly, we need to be able to tell whether a gate is useless, and secondly whenever we remove a useless gate, we need to "re-wire" the gates that come after that gate, which can depend on the types of the useless gate that we are removing, as described in Proposition 6.

More specifically, DS will be a "tree-like" structure of depth at most $\ell \cdot (\ell - 1)/2 + 2$ where the internal nodes are labelled as gates. Note that a path from the root to any internal node in the tree gives a sequence of gates, which specifies a comparator circuit up to a choice of the output wire. We will require the label of every internal node to be a useful gate in the circuit specified by the path from the root to the node. In other words, each internal node will branch on all possible useful gates that could occur next in the circuit. Moreover, each leaf is either labelled as a useless gate, with respect to the circuit specified by the path from the root to the current leaf, or is labelled as a single wire that is designed to be the output wire.

For every leaf that is a useless gate, we store its type, and for each leaf that is a single wire, we store the number of satisfying assignments of the circuit that is specified by the path from the root to the leaf. Moreover, each internal node is a useful gate whose children are indexed by the set of all possible gates (each is an ordered pair of wires) and the set of wires (called an *output leaf*). Note that checking whether a new gate is useless and computing its type require evaluating the current circuit on all possible inputs, which takes time $2^n \cdot \text{poly}(\ell)$, but this is fine with our running time. Similarly, we can compute the number of satisfying assignments in each output leaf by brute force. Note that by Theorem 7, the depth of such a tree is at most $\ell \cdot (\ell - 1)/2 + 2$, otherwise there would be a comparator circuit with ℓ wires that has more than $\ell \cdot (\ell - 1)/2$ useful gates. Since each internal node has at most ℓ^2 children, the tree has at most $\ell^{O(\ell^2)}$ nodes in total. Since each node can be constructed in time $2^n \cdot \text{poly}(\ell)$, the running time is clear.

To look up the number of the satisfying assignments of a given circuit C (with a labelling of the wires that is consistent with the one used for DS), we start from the root of DS, and move down the tree as we look at the gates in C one by one (from left to right in the natural way). If we reach an output leaf, we output the number of satisfying assignments stored in that leaf. However, if we reach a leaf v that is specified as a useless gate, we remove the corresponding gate in C and update the gates that come after it according to the type of this useless gate, using Proposition 6. Once we update the circuit, we start again from the parent of v and look at the next gate in the updated circuit. We repeat this until we reach an output leaf. □

5.2 The Algorithm

We will show an algorithm for comparator circuits with small number of wires, while the number of gates can be polynomial.

Theorem 14 *There is a constant $d > 1$ and a deterministic algorithm such that for every $k \leqslant n/4$, given a comparator circuit on n variables with at most*

$$\frac{n^{1.5}}{d \cdot k \cdot \sqrt{\log n}}$$

wires and $\mathrm{poly}(n)$ *gates, the algorithm outputs the number of satisfying assignments of C in time*

$$2^{n-k} \cdot \mathrm{poly}(n).$$

Proof Let C be a comparator circuit with $\ell \leqslant n^{1.5}/\left(d \cdot k \cdot \sqrt{\log n}\right)$ wires and $\mathrm{poly}(n)$ gates, where $d \geqslant 1$ is a sufficiently large constant.

We partition the n variables almost-evenly into $\lfloor n/k \rfloor$ consecutive blocks, denoted as $S_1, S_2, \ldots, S_{\lfloor n/k \rfloor}$. We then count the number of wires that are labelled by the variables in each S_i and let

$$w_i := \left|\{u : u \text{ is a wire labelled by some } x \in S_i \text{ (or its negation)}\}\right|.$$

We have

$$\sum_{i \in [\lfloor n/k \rfloor]} w_i = \ell,$$

which implies that there is a particular $i \in [k]$ such that

$$w_i \leqslant \frac{\ell}{\lfloor n/k \rfloor} \leqslant \frac{1}{d} \cdot \sqrt{\frac{n}{\log n}} =: \ell_0.$$

Moreover, we can find such i efficiently.

Constructing DS Using Lemma 13, we create a data structure DS with w_i wires and $|S_i| \leqslant k$ variables and a labelling consistent with that of C for the wires labelled by variables from S_i. This can be done in time

$$2^k \cdot \ell_0^{O(\ell_0^2)} = 2^{k + O(\ell_0^2 \cdot \log \ell_0)} \leqslant 2^{k+n/2}.$$

Enumeration For each $\rho \in \{0,1\}^{[n] \setminus S_i}$, we obtain a restricted circuit $C\restriction_\rho$ (on either k or $k+1$ variables), which has ℓ_0 wires (whose labelling is consistent with the one used for DS created above) and has $\mathrm{poly}(n)$ gates. Then using DS, we can efficiently look up the number of satisfying assignments of $C\restriction_\rho$. Finally we sum over these numbers over all such ρ's and this gives the number of satisfying assignments of C.

The total running time of the above algorithm is

$$2^{k+n/2} + 2^{n-k} \cdot \text{poly}(n) = 2^{n-k} \cdot \text{poly}(n),$$

as desired. □

6 Pseudorandom Generators and MCSP Lower Bounds

In this section, we show a PRG for small comparator circuits, and derive from it lower bounds for comparator circuits computing MCSP.

6.1 Proof of the PRG

We start with some definitions and notations.

- For a Boolean function f, we denote by $\ell(f)$ the minimum number of wires in a comparator circuit computing f.
- We will often describe a restriction $\rho \in \{0, 1, *\}^n$ as a pair $(\sigma, \beta) \in \{0, 1\}^n \times \{0, 1\}^n$. The string σ is the characteristic vector of the set of coordinates that are assigned $*$ by ρ, and β is an assignment of values to the remaining coordinates. The string σ is also called a *selection*.
- We say that a distribution $\mathcal{D}$ on $\{0, 1\}^n$ is a *p-regular* random *selection* if $\mathop{\mathbf{Pr}}_{\sigma\sim\mathcal{D}}[\sigma(i) = 1] = p$ for every $i \in [n]$.

As mentioned in Sect. 1.2, we will need a result saying that the number of wires in a comparator circuit shrinks with high probability under pseudorandom restrictions.

Lemma 15 *Let c be a constant and let $f : \{0, 1\}^n \to \{0, 1\}$. Let $\ell := \ell(f)$ and $p = \ell^{-2/3}$, and suppose that $\ell = n^{\Omega(1)}$. There exists a p-regular pseudorandom selection $\mathcal{D}$ over n variables that is samplable using $r = \text{polylog}(\ell)$ random bits such that*

$$\mathop{\mathbf{Pr}}_{\sigma\sim\mathcal{D},\, \beta\sim\{0,1\}^n}\left[\ell(f\restriction_{(\sigma,\beta)}) \geqslant 2^{3\sqrt{c\log\ell}} \cdot p\ell\right] \leqslant 2 \cdot \ell^{-c}.$$

Moreover, there exists a circuit of size $\text{polylog}(\ell)$ *such that, given $j \in \{0, 1\}^{\log n}$ and a seed $z \in \{0, 1\}^r$, the circuit computes the jth coordinate of $\mathcal{D}(z)$.*

The proof of Lemma 15 follows closely that of [16, Lemma 5.3], except for that here we also need to show that the pseudorandom restriction can be computed with small size circuits. Such a restriction is proved to exist in Lemma 18 of [28]. For completeness, a proof is presented in "Appendix B".

Theorem 16 (Local PRGs) *For every $n \in \mathbb{N}$, $\ell = n^{\Omega(1)}$, and $\varepsilon \geqslant 1/\text{poly}(n)$, there is a pseudorandom generator $G\colon \{0, 1\}^r \to \{0, 1\}^n$, with seed length*

$$r = \ell^{2/3+o(1)}$$

that ε-fools comparator circuits with ℓ wires over n variables. Moreover, for every seed $z \in \{0,1\}^r$, there is a circuit D_z of size $\ell^{2/3+o(1)}$ such that, given as input $j \in [n]$, D_z computes the jth bit of $G(z)$.

Proof Sketch In [16], it is shown that if a circuit class "shrinks" with high probability under a pseudorandom restriction, then we can construct pseudorandom generators for this circuit class with non-trivial seed-length. The authors of [28] then showed that if the same shrinkage property holds for random selections that can be efficiently sampled and computed, then we can obtain local PRGs. In Lemma 15, we proved exactly what is required by [28] to obtain local PRGs for comparator circuits.

More specifically, the theorem can be derived by following the proof of [28, Lemma 16], and adjusting the parameters there in a natural way. In particular, we will use $p := \ell^{-2/3}$ so that after the pseudorandom restriction in Lemma 15, the restricted comparator circuit has at most $\ell_0 := 2^{O(\sqrt{\log \ell})} \cdot p\ell = 2^{O(\sqrt{\log \ell})} \cdot \ell^{1/3}$ wires (with high probability). Another observation needed in the proof is that, by Lemma 8, there can be at most $2^{\ell^{2/3+o(1)}}$ distinct functions for comparator circuits with this many wires. We omit the details here. □

6.2 Proof of the MCSP Lower Bound

We prove the following stronger result which implies Theorem 4.

Theorem 17 *For any $\varepsilon > 0$ and any $0 < \alpha \leqslant 1 - \varepsilon$, $\mathsf{MCSP}[n^\alpha]$ on inputs of length n cannot be computed by comparator circuits with $n^{1+\alpha/2-\varepsilon}$ wires.*

Proof Let f denote the function $\mathsf{MCSP}[n^\alpha]$ on inputs of length n. For the sake of contradiction, suppose f can be computed by a comparator circuit C with $n^{1+\alpha/2-\varepsilon}$ wires, for some $\varepsilon > 0$.

Let $k := n^{\alpha+\varepsilon/2}$. Consider an (almost-even) partition of the n variables into $\lfloor n/k \rfloor$ *consecutive* blocks, denoted as $S_1, S_2, \ldots, S_{\lfloor n/k \rfloor}$. Again, by an averaging argument, there is some $i \in [\lfloor n/k \rfloor]$ such that after fixing the values of the variables outside S_i, the number of wires in the restricted circuit is at most

$$\ell := n^{1+\alpha/2-\varepsilon} / \lfloor n/k \rfloor = n^{1.5\alpha-\varepsilon/2}.$$

Let ρ be a restriction that fixes the values of the variables outside S_i to be 0 and leaves the variables in S_i unrestricted. Let G be the PRG from Theorem 16 that has seed length $r := \ell^{2/3+o(1)}$ and $(1/3)$-fools comparator circuits with at most ℓ wires.

On the one hand, since $|S_i| \geqslant k$, then by a counting argument, for a uniformly random $x \in \{0,1\}^n$, the circuit size of the truth table given by $\rho \circ x$ is at least $k/(10 \log k) > n^\alpha$, with probability at least $1/2$. In other words,

$$\mathop{\mathbf{Pr}}_{x \in \{0,1\}^n} [f\restriction_\rho (x) = 1] \leqslant 1/2.$$

On the other hand, by the second item of Theorem 16, for any seed $z \in \{0,1\}^r$, the output of the PRG $G(z)$, viewed as a truth table, represents a function that can be

computed by a circuit of size $\ell^{2/3+o(1)}$. Then knowing $i \in [n]$ (which can be encoded using $\log(n)$ bits), the truth table given by $\rho \circ G(z)$ has circuit size at most

$$\text{polylog}(n) + \ell^{2/3+o(1)} \leqslant n^{\alpha}.$$

This implies

$$\mathop{\mathbf{Pr}}_{z\in\{0,1\}^r}[C\restriction_{\rho}(G(z)) = 1] = 1,$$

which contradicts the security of G. □

7 Learning Algorithms

Recall that a (distribution-independent) PAC learning algorithm for a class of functions $\mathcal{C}$ has access to labelled examples $(x, f(x))$ from an unknown function $f \in \mathcal{C}$, where x is sampled according to some (also unknown) distribution $\mathcal{D}$. The goal of the learner is to output, with high probability over its internal randomness and over the choice of random examples, a hypothesis h that is close to f under $\mathcal{D}$. As in [18], here we consider the stronger model of "randomized exact learning from membership and equivalence queries". It is known that learnability in this model implies learnability in the distribution-independent PAC model with membership queries (see [18, Sect. 2] and the references therein).

Theorem 18 [18, Lemma 4.4] *Fix any partition $S_1, S_2, \ldots, S_{n^{1-n^{\delta}}}$ of $[n]$ into equal-size subsets, where each S_i is of size n^{δ} and $\delta > 0$. Let $\mathcal{C}$ be a class of n-variate functions such that for each $f \in \mathcal{C}$, there is an S_i such that $\left|\left\{f\restriction_{\rho}\right\}_{\rho\in\{0,1\}^{[n]\setminus S_i}}\right| \leqslant 2^{n^{\beta}}$, where $\beta < 1$ and moreover $\delta + \beta < 1$. Then there is a randomized exact learning algorithms for $\mathcal{C}$ that uses membership and equivalence queries and runs in time $2^{n-n^{\delta}} \cdot \text{poly}(n)$.*

Corollary 19 *For every $\varepsilon > 0$, there is a randomized exact learning algorithms for comparator circuits with $n^{1.5-\varepsilon}$ wires that uses membership and equivalence queries that runs in time $2^{n-n^{\Omega(\varepsilon)}} \cdot \text{poly}(n)$.*

Proof Consider Theorem 18 and any partition $S_1, S_2, \ldots, S_{n^{1-n^{\delta}}}$ of the n variables into equal-size subsets, each is of size n^{δ}, where $\delta := \varepsilon/3$. Then by an averaging argument, for every comparator circuit C with $n^{1.5-\varepsilon}$ wires, there is some S_i such that after fixing the variables outside of S_i, the number of wires in the restricted circuit is at most $\ell := n^{1.5-\varepsilon}/n^{1-\delta} \leqslant n^{.5-2\varepsilon/3}$. By Lemma 8, such a restricted circuit computes some function that is equivalent to a circuit with $\ell(\ell-1)/2$ gates, and there are at most $\ell^{O(\ell^2)} \leqslant 2^{n^{1-\varepsilon/2}}$ such circuits. Therefore we have

$$\left|\left\{C\restriction_{\rho}\right\}_{\rho\in\{0,1\}^{[n]\setminus S_i}}\right| \leqslant 2^{n^{\beta}},$$

where $\beta := 1-\varepsilon/2 < 1$ and $\delta+\beta < 1$. The algorithm then follows from Theorem 18. □

Acknowledgements B. P. Cavalar acknowledges support of the Chancellor's International Scholarship of the University of Warwick. Z. Lu acknowledges support from the Royal Society University Research Fellowship URF\R1\191059. Part of this work was done while Z. Lu was a Research Fellow at the University of Warwick. Both authors are indebted to Igor C. Oliveira for numerous helpful discussions and comments. We also thank the anonymous reviewers of ITCS 2022 for comments and suggestions.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

A Proof of Theorem 12

The hard function We need to slightly modify the hard function in Definition 10 (particularly the function α) to adjust an arbitrary partition as in Theorem 12. For an integer k and a partition of n variables into n/k equal-sized blocks, denoted by $S := \{S_1, S_2, \ldots, S_{n/k}\}$, define $A_{S,k}\colon \{0,1\}^{n+n} \to \{0,1\}$ as follows:

$$A_{S,k}(x_1,\ldots,x_n,y_1,\ldots,y_n) := \mathrm{Enc}(x_1,\ldots,x_n)_{\alpha(y_1,\ldots,y_n)},$$

where Enc is the code from Theorem 9 that maps n bits to 2^k bits, and $\alpha\colon \{0,1\}^n \to \{0,1\}^k$ is defined as

$$\alpha(y_1,\ldots,y_n) := \left(\bigoplus_{z\in B_1} z, \bigoplus_{z\in B_2} z, \ldots, \bigoplus_{z\in B_k} z\right),$$

where $B_j := \bigcup_{i\in[n/k]} \{z\colon z \text{ is the } j\text{th variables of } S_i\}$.

Good x We will need the following lemma which says that for most $x \in \{0,1\}^n$, the codeword of x is hard to approximate for any fixed small set of functions.

Lemma 20 *Let k be such that $c\cdot\log n \leqslant k \leqslant n/6$, where c is the constant from Theorem 9, and let* Enc *be the code from Theorem 9 that maps n bits to 2^k bits. Let $\mathcal{H}'$ be a set of k-input-bit Boolean functions such that $|\mathcal{H}'| \leqslant 2^{2n/3}$. Then, with probability at least $1-1/2^{n/2}$ over a random $x \in \{0,1\}^n$, the following holds for every $f \in \mathcal{H}'$:*

$$\Pr_{z\in\{0,1\}^k}\left[f(z) = \mathrm{Enc}(x)_z\right] \leqslant \frac{1}{2} + \frac{n}{2^{k/4}}. \tag{2}$$

Proof The proof is by a counting argument. For every $f \in \mathcal{H}'$, consider the 2^k-bit string $\mathrm{tt}(f)$ which is the truth table computed by f. Let us say x is bad for f if (2) does not hold, which means that Equation $\mathrm{tt}(f)$ and $\mathrm{Enc}(x)$ agree on more than $1/2 + n/2^{k/4}$ positions. By the list-decodability of Enc, the number of such x's is at most $O\left(2^{k/2}/n\right)$. By an union bound over all the $2^{2n/3}$ functions in $\mathcal{H}'$, the fraction of bad x's is at most

$$\frac{O\left(2^{k/2}/n\right) \cdot 2^{2n/3}}{2^n} < \frac{1}{2^{n/2}},$$

as desired. □

We are now ready to prove Theorem 12.

Proof of Theorem 12 Let $A := A_{S,k}$ be the hard function on $2n$ variables defined as above, where S is the partition in the statement of the theorem, and let

$$B_j := \bigcup_{i \in [n/k]} \{z \colon z \text{ is the } j\text{th variables of } S_i\}.$$

Also, let $\mathcal{H}'$ be the set of k-input-bit Boolean functions defined as follows:

$$\mathcal{H}' := \left\{ f \colon \exists h \in \mathcal{H} \text{ and } w \in \{0,1\}^k, \text{ such that } f(z) = h(z \oplus w) \text{ for all } z \in \{0,1\}^k \right\}.$$

That is, $\mathcal{H}'$ is the set of all possible "shifted" functions in $\mathcal{H}$. By Lemma 20, with probability at least $1 - 1/2^{n/2}$ over a random $x \in \{0,1\}^n$, for every $f \in \mathcal{H}'$ we have

$$\mathop{\mathbf{Pr}}_{z \in \{0,1\}^k} \left[f(z) = \mathrm{Enc}(x)_z \right] \leqslant \frac{1}{2} + \frac{n}{2^{k/4}}. \tag{3}$$

Let us call x *good* if it satisfies Eq. (3).

To show the theorem, we need to upper bound the following probability, for every circuit $C_0 \in \mathcal{C}_{2n}$ of size $s(n,k)$:

$$\begin{aligned}
&\mathop{\mathbf{Pr}}_{x,y \in \{0,1\}^n \times \{0,1\}^n} [A(x,y) = C_0(x,y)] \\
&\quad \leqslant \mathop{\mathbf{Pr}}_{x,y} [A(x,y) = C_0(x,y) \mid x \text{ is good}] + \mathop{\mathbf{Pr}}_{x} [x \text{ is not good}] \\
&\quad \leqslant \mathop{\mathbf{Pr}}_{x,y} [A(x,y) = C_0(x,y) \mid x \text{ is good}] + \frac{1}{2^{n/2}}.
\end{aligned}$$

Let x be any fixed n-bit string that is good. Let $A' \colon \{0,1\}^n \to \{0,1\}$ be

$$A'(y) := A(x,y),$$

and let C be the circuit defined as

$$C(y) := C_0(x,y).$$

Note that since the class $\mathcal{C}$ is closed under restriction, C is a circuit from $\mathcal{C}_n$ with size at most $s(n,k)$. We will show that

$$\Pr_{y\in\{0,1\}^n}\left[A'(y)=C(y)\right]\leqslant\frac{1}{2}+\frac{n}{2^{k/4}}.$$

Let S_i be the block in the assumption of the theorem such that

$$\left\{C\restriction_\rho\right\}_{\rho\in\{0,1\}^{[n]\setminus S_i}}\subseteq\mathcal{H}.$$

We have

$$\Pr_{y\in\{0,1\}^n}\left[A'(y)=C(y)\right]=\Pr_{\rho\in\{0,1\}^{[n]\setminus S_i},z\in\{0,1\}^k}\left[A'\restriction_\rho(z)=C\restriction_\rho(z)\right].$$

It suffices to upper bound

$$\Pr_{z\in\{0,1\}^k}\left[A'\restriction_\rho(z)=C\restriction_\rho(z)\right]$$

for every $\rho\in\{0,1\}^{[n]\setminus S_i}$. For the sake of contradiction, suppose for some ρ, we have

$$\frac{1}{2}+\frac{n}{2^{k/4}}<\Pr_{z\in\{0,1\}^k}\left[A'\restriction_\rho(z)=C\restriction_\rho(z)\right]=\Pr_{z\in\{0,1\}^k}\left[\mathrm{Enc}(x)_\alpha=C\restriction_\rho(z)\right],\quad(4)$$

where $\alpha\in\{0,1\}^k$ is

$$\alpha_j:=\mathsf{Parity}\left(\rho\restriction_{B_j\setminus S_i}\right)\oplus z_j,$$

and $\rho\restriction_{B_j\setminus S_i}$ denotes the partial assignment given by ρ but restricted to only variables in the set $B_j\setminus S_i$. That is, α is some "shift" of z, so α is uniformly distributed for uniformly random z. Therefore, Eq. (4) implies

$$\Pr_{z\in\{0,1\}^k}\left[\mathrm{Enc}(x)_z=C\restriction_\rho(z\oplus w)\right]>\frac{1}{2}+\frac{n}{2^{k/4}},$$

for some $w\in\{0,1\}^k$. This gives a function in $\mathcal{H}'$ that computes $\mathrm{Enc}(x)$ on more than $1/2+n/2^{k/4}$ positions, which contradicts the assumption that x is good. □

B Pseudorandom Shrinkage for Comparator Circuits: Proof of Lemma 15

Technical tools We will need a Chernoff-Hoeffding bounds for distributions with bounded independence from [32] (Lemmas 2.3 in [16]). Recall that a distribution $\mathcal{D}$ on $[m]^n$ is *k-wise independent* if, for any set $A\subseteq[n]$ of size $|A|\leqslant k$, the random variables $\{\sigma(i):i\in A\}$ are mutually independent when $\sigma\sim\mathcal{D}$.

Lemma 21 [32] *Let $a_1, \ldots, a_n \in \mathbb{R}_+$ and let $m = \max_i a_i$. Suppose that $X_1, \ldots, X_n \in \{0,1\}$ are k-wise independent random variables with $\mathbf{Pr}[X_i = 1] = p$. Let $X = \sum_i a_i X_i$ and $\mu = \mathbf{E}[X] = p \sum_i a_i$. We have $\mathbf{Pr}[X \geqslant 2k(m+\mu)] \leqslant 2^{-k}$.*

Lemma 22 [16, Lemma 2.4] *Let $X_1, \ldots, X_n \in \{0,1\}$ be k-wise independent random variables with $\mathbf{Pr}[X_i = 1] = p$. Let $X = \sum_i X_i$ and $\mu = \mathbf{E}[X] = np$. We have $\Pr[X \geqslant k] \leqslant \mu^k/k!$.*

Shrinkage of comparator circuits under pseudorandom restrictions We first show the following result for comparator circuits which is analogous to [16, Lemma 5.2] for branching programs.

Lemma 23 *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function, and let $H \subseteq [n]$. For $h \in \{0,1\}^H$, let ρ_h denote the restriction that sets the variables in H to h, and leaves the other variables free. We have $\ell(f) \leqslant 2^{|H|} \cdot \left(\max_{h \in \{0,1\}^H} \ell(f\restriction_{\rho_h}) + |H|\right)$.*

Proof For $h \in \{0,1\}^H$, let $\mathbb{1}_h : x \mapsto \mathbb{1}\{x = h\}$. Clearly, $\mathbb{1}_h$ can be computed by a comparator circuit with $|H|$ wires. Since $f = \bigvee_{h \in \{0,1\}^H} (\mathbb{1}_h \wedge f\restriction_{\rho_h})$, the result follows. □

Lemma 24 (Reminder of Lemma 15) *Let c be a constant and let $f : \{0,1\}^n \to \{0,1\}$. Let $\ell := \ell(f)$ and $p = \ell^{-2/3}$, and suppose that $\ell = n^{\Omega(1)}$. There exists a p-regular pseudorandom selection $\mathcal{D}$ over n variables that is samplable using $r = \mathrm{polylog}(\ell)$ random bits such that*

$$\mathop{\mathbf{Pr}}_{\sigma \sim \mathcal{D},\, \beta \sim \{0,1\}^n} \left[\ell(f\restriction_{(\sigma,\beta)}) \geqslant 2^{3\sqrt{c \log \ell}} \cdot p\ell\right] \leqslant 2 \cdot \ell^{-c}.$$

Moreover, there exists a circuit of size $\mathrm{polylog}(\ell)$ such that, given $j \in \{0,1\}^{\log n}$ and a seed $z \in \{0,1\}^r$, the circuit computes the jth coordinate of $\mathcal{D}(z)$.

Proof First, we note that a k-wise independent random selection that can be efficiently sampled and computed with the required parameters is proved to exist in Lemma 18 of [28]. Henceforth, we let ρ be the random restriction described by the pair (σ, β).

Let C be a comparator circuit with ℓ wires computing f. Let $k = c \cdot \log \ell$. For $i \in [n]$, let w_i be the number of wires in C labelled with the variable x_i.

Let $\alpha = \sqrt{c/\log \ell}$. We say that $i \in [n]$ is *heavy* if $w_i \geqslant p^{1-\alpha} \cdot \ell$ and *light* otherwise. Let $H \subseteq [n]$ be the set of heavy variables. We have $|H| \leqslant (1/p)^{1-\alpha}$. Let also $H(\rho) := H \cap \rho^{-1}(*)$. Let ρ' be a restriction such that $\rho'(x) = \rho(x)$ for $x \notin H(\rho)$ and which sets the variables in $H(\rho)$ so as to maximize $\ell(f\restriction_{\rho'})$. By Lemma 23, we have $\ell(f\restriction_{\rho}) \leqslant 2^{|H(\rho)|+1} \cdot \ell(f\restriction_{\rho'})$.

We now let $h = \lceil 3/2 \cdot c/\alpha \rceil$, and observe that

$$\mathop{\mathbf{Pr}}_{\rho}\left[\ell(f\restriction_{\rho}) \geqslant 2^{h+3} k p^{1-\alpha} s\right] \leqslant \mathop{\mathbf{Pr}}_{\rho}\left[|H(\rho)| \geqslant h\right] + \mathop{\mathbf{Pr}}_{\rho}\left[\ell(f\restriction_{\rho'}) \geqslant 4kp^{1-\alpha}\ell\right].$$

Let X_i be a random variable such that $X_i = 1$ iff $\rho(i) = *$. From Lemma 22, it follows that the first term can be bounded by $(|H|p)^h \leqslant p^{\alpha h} \leqslant \ell^{-c}$. For the second term, we can apply Lemma 21 on the light variables with $\mu \leqslant p\ell$ and $m < p^{1-\alpha}\ell$, so that $m + \mu \leqslant 2p^{1-\alpha}\ell$, thus bounding the probability by $2^{-k} \leqslant \ell^{-c}$. □

References

1. Hrapčenko, V.M.: A certain method of obtaining estimates from below of the complexity of π-schemes. Mat. Zametki **10**, 83–92 (1971)
2. Mayr, E.W., Subramanian, A.: The complexity of circuit value and network stability. J. Comput. Syst. Sci. **44**(2), 302–323 (1992). https://doi.org/10.1016/0022-0000(92)90024-D
3. Cook, S.A., Filmus, Y., Le, D.T.M.: The complexity of the comparator circuit value problem. ACM Trans. Comput. Theory **6**(4), 15–11544 (2014). https://doi.org/10.1145/2635822
4. Robere, R., Pitassi, T., Rossman, B., Cook, S.A.: Exponential lower bounds for monotone span programs. In: Symposium on Foundations of Computer Science (FOCS), pp. 406–415 (2016)
5. Ajtai, M., Komlós, J., Szemerédi, E.: An $O(n \log n)$ sorting network. In: Symposium on Theory of Computing (STOC), pp. 1–9 (1983). https://doi.org/10.1145/800061.808726
6. Razborov, A.A.: Lower bounds on the complexity of realization of symmetric Boolean functions by gate switching circuits. Mat. Zametki **48**(6), 79–90 (1990). https://doi.org/10.1007/BF01240265
7. Göös, M., Kamath, P., Robere, R., Sokolov, D.: Adventures in monotone complexity and TFNP. In: Innovations in Theoretical Computer Science Conference (ITCS), pp. 38–13819 (2019)
8. Komarath, B., Sarma, J., Sunil, K.S.: Comparator circuits over finite bounded posets. Inf. Comput. **261**, 160–174 (2018). https://doi.org/10.1016/j.ic.2018.02.002
9. Gál, A., Robere, R.: Lower bounds for (non-monotone) comparator circuits. In: Innovations in Theoretical Computer Science Conference (ITCS), pp. 58–15813 (2020)
10. Nechiporuk, E.I.: On a Boolean function. Dokl. Akad. Nauk SSSR **169**, 765–766 (1966)
11. Oliveira, I.C.: Algorithms versus circuit lower bounds. CoRR arXiv:1309.0249 (2013). Accessed 01 Sep 2021
12. Williams, R.: Algorithms for circuits and circuits for algorithms. In: Conference on Computational Complexity (CCC), pp. 248–261 (2014)
13. Williams, R.: Nonuniform ACC circuit lower bounds. J. ACM **61**(1), 2–1232 (2014)
14. Impagliazzo, R., Matthews, W., Paturi, R.: A satisfiability algorithm for AC^0. In: Symposium on Discrete Algorithms (SODA), pp. 961–972 (2012)
15. Tal, A.: #SAT algorithms from shrinkage. Electron. Colloq. Comput. Complex. 114 (2015)
16. Impagliazzo, R., Meka, R., Zuckerman, D.: Pseudorandomness from shrinkage. J. ACM **66**(2), 11–11116 (2019). https://doi.org/10.1145/3230630
17. Chen, R., Kabanets, V., Kolokolova, A., Shaltiel, R., Zuckerman, D.: Mining circuit lower bound proofs for meta-algorithms. Comput. Complex. **24**(2), 333–392 (2015). https://doi.org/10.1007/s00037-015-0100-0
18. Servedio, R.A., Tan, L.: What circuit classes can be learned with non-trivial savings? In: Innovations in Theoretical Computer Science Conference (ITCS), pp. 30–13021 (2017)
19. Kabanets, V., Koroth, S., Lu, Z., Myrisiotis, D., Oliveira, I.C.: Algorithms and lower bounds for De Morgan formulas of low-communication leaf gates. In: Conference on Computational Complexity (CCC), pp. 15–11541 (2020)
20. Komargodski, I., Raz, R.: Average-case lower bounds for formula size. In: Symposium on Theory of Computing (STOC), pp. 171–180 (2013). https://doi.org/10.1145/2488608.2488630
21. Komargodski, I., Raz, R., Tal, A.: Improved average-case lower bounds for De Morgan formula size: matching worst-case lower bound. SIAM J. Comput. **46**(1), 37–57 (2017). https://doi.org/10.1137/15M1048045
22. Williams, R.: Improving exhaustive search implies superpolynomial lower bounds. SIAM J. Comput. **42**(3), 1218–1244 (2013)
23. Oliveira, I.C., Santhanam, R.: Hardness magnification for natural problems. In: Symposium on Foundations of Computer Science (FOCS), pp. 65–76 (2018). https://doi.org/10.1109/FOCS.2018.00016
24. Oliveira, I.C., Pich, J., Santhanam, R.: Hardness magnification near state-of-the-art lower bounds. In: Computational Complexity Conference (CCC), pp. 27–12729 (2019). https://doi.org/10.4230/LIPIcs.CCC.2019.27
25. Chen, L., Jin, C., Williams, R.R.: Hardness magnification for all sparse NP languages. In: Symposium on Foundations of Computer Science (FOCS), pp. 1240–1255 (2019). https://doi.org/10.1109/FOCS.2019.00077
26. Chen, L., Hirahara, S., Oliveira, I.C., Pich, J., Rajgopal, N., Santhanam, R.: Beyond natural proofs: hardness magnification and locality. In: Innovations in Theoretical Computer Science Conference (ITCS), pp. 70–17048 (2020). https://doi.org/10.4230/LIPIcs.ITCS.2020.70

27. Golovnev, A., Ilango, R., Impagliazzo, R., Kabanets, V., Kolokolova, A., Tal, A.: $AC^0[p]$ lower bounds against MCSP via the coin problem. In: International Colloquium on Automata, Languages, and Programming (ICALP), pp. 66–16615 (2019)
28. Cheraghchi, M., Kabanets, V., Lu, Z., Myrisiotis, D.: Circuit lower bounds for MCSP from local pseudorandom generators. ACM Trans. Comput. Theory **12**(3), 21–12127 (2020). https://doi.org/10.1145/3404860
29. Håstad, J.: Almost optimal lower bounds for small depth circuits. In: Symposium on Theory of Computing (STOC), pp. 6–20 (1986). https://doi.org/10.1145/12130.12132
30. Chen, L., Jin, C., Williams, R.R.: Sharp threshold results for computational complexity. In: Symposium on Theory of Computing (STOC), pp. 1335–1348 (2020). https://doi.org/10.1145/3357713.3384283
31. Jukna, S.: Boolean Function Complexity—Advances and Frontiers. Algorithms and Combinatorics, vol. 27. Springer, Berlin (2012). https://doi.org/10.1007/978-3-642-24508-4
32. Schmidt, J.P., Siegel, A., Srinivasan, A.: Chernoff–Hoeffding bounds for applications with limited independence. SIAM J. Discrete Math. **8**(2), 223–250 (1995). https://doi.org/10.1137/S089548019223872X

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.