Lower Bounds Against Sparse Symmetric Functions of ACC Circuits: Expanding the Reach of #SAT Algorithms

Nikhil Vyas
MIT, Cambridge, MA, USA
nikhilv@mit.edu

R. Ryan Williams
MIT, Cambridge, MA, USA
rrw@mit.edu

Abstract

We continue the program of proving circuit lower bounds via circuit satisfiability algorithms. So far, this program has yielded several concrete results, proving that functions in Quasi-NP = NTIME$[n^{(\log n)^{O(1)}}]$ and NEXP do not have small circuits (in the worst case and/or on average) from various circuit classes $\mathcal{C}$, by showing that $\mathcal{C}$ admits non-trivial satisfiability and/or #SAT algorithms which beat exhaustive search by a minor amount.

In this paper, we present a new strong lower bound consequence of non-trivial #SAT algorithm for a circuit class $\mathcal{C}$. Say a symmetric Boolean function $f(x_1, \ldots, x_n)$ is *sparse* if it outputs 1 on $O(1)$ values of $\sum_i x_i$. We show that for every sparse f, and for all "typical" $\mathcal{C}$, faster #SAT algorithms for $\mathcal{C}$ circuits actually imply lower bounds against the circuit class $f \circ \mathcal{C}$, which may be *stronger* than $\mathcal{C}$ itself. In particular:

- #SAT algorithms for n^k-size $\mathcal{C}$-circuits running in $2^n/n^k$ time (for all k) imply NEXP does not have $f \circ \mathcal{C}$-circuits of polynomial size.
- #SAT algorithms for 2^{n^ε}-size $\mathcal{C}$-circuits running in 2^{n-n^ε} time (for some $\varepsilon > 0$) imply Quasi-NP does not have $f \circ \mathcal{C}$-circuits of polynomial size.

Applying #SAT algorithms from the literature, one immediate corollary of our results is that Quasi-NP does not have EMAJ $\circ$ ACC^0 $\circ$ THR circuits of polynomial size, where EMAJ is the "exact majority" function, improving previous lower bounds against ACC^0 [Williams JACM'14] and ACC^0 $\circ$ THR [Williams STOC'14], [Murray-Williams STOC'18]. This is the first nontrivial lower bound against such a circuit class.

2012 ACM Subject Classification Theory of computation → Circuit complexity

Keywords and phrases #SAT, satisfiability, circuit complexity, exact majority, ACC

Digital Object Identifier 10.4230/LIPIcs.STACS.2020.59

Related Version A full version of the paper is available at `https://arxiv.org/abs/2001.07788`.

Funding Supported by NSF CCF-1741615 and NSF CCF-1909429.

1 Introduction

Currently, our knowledge of algorithms vastly exceeds our knowledge of lower bounds. Is it possible to bridge this gap, and use the existence of powerful algorithms to give lower bounds for hard functions? Over the last decade, the program of proving lower bounds via algorithms has been positively addressing this question. A line of work starting with Kabanets and Impagliazzo [15] has shown how deterministic subexponential-time algorithms for polynomial identity testing would imply lower bounds against arithmetic circuits. Starting around 2010 [24, 25], it was shown that even *slightly nontrivial* algorithms could imply Boolean circuit lower bounds. For example, a circuit satisfiability algorithm running in $O(2^n/n^k)$

37th International Symposium on Theoretical Aspects of Computer Science (STACS 2020).
Editors: Christophe Paul and Markus Bläser; Article No. 59; pp. 59:1–59:17

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

time (for all k) on n^k-size circuits with n inputs would already suffice to yield the (infamously open) lower bound $\mathsf{NEXP} \not\subset \mathsf{P/poly}$. More generally, a generic connection was found between non-trivial SAT algorithms and circuit lower bounds:

▶ **Theorem 1** ([24, 25], Informal). *Let $\mathcal{C}$ be a circuit class closed under AND, projections, and compositions.*[1] *Suppose for all k there is an algorithm A such that, for every $\mathcal{C}$-circuit of n^k size, A determines its satisfiability in $O(2^n/n^k)$ time. Then* NEXP *does not have polynomial-size $\mathcal{C}$-circuits.*

To illustrate Theorem 1 with two examples, when $\mathcal{C}$ is the class of general fan-in 2 circuits, Theorem 1 says that non-trivial Circuit SAT algorithms imply $\mathsf{NEXP} \not\subset \mathsf{P/poly}$; when $\mathcal{C}$ is the class of Boolean formulas, it says non-trivial Formula-SAT algorithms imply $\mathsf{NEXP} \not\subset \mathsf{NC}^1$. Both are major open questions in circuit complexity. Theorem 1 and related results have been applied to prove several concrete circuit lower bounds: super-polynomial lower bounds for ACC^0 [25], $\mathsf{ACC}^0 \circ \mathrm{THR}$ [21], quadratic lower bounds for depth-two symmetric and threshold circuits [18, 1], and average-case lower bounds as well [7, 5].

Recently, the algorithms-to-lower-bounds connection has been extended to show a trade-off between the running time of the SAT algorithm on large circuits, and the complexity of the hard function in the lower bound. In particular, it is even possible in principle to obtain circuit lower bounds against NP with this algorithmic approach.

▶ **Theorem 2** ([16], Informal). *Let $\mathcal{C}$ be a class of circuits closed under unbounded AND, ORs of fan-in two, and negation. Suppose there is an algorithm A and $\varepsilon > 0$ such that, for every $\mathcal{C}$-circuit C of 2^{n^ε} size, A solves satisfiability for C in $O(2^{n-n^\varepsilon})$ time. Then* Quasi-NP *does not have polynomial-size $\mathcal{C}$-circuits.*[2]

In fact, Theorem 2 holds even if A only distinguishes between unsatisfiable circuits from those with at least 2^{n-1} SAT assignments; we call this easier problem GAP-UNSAT.

Intuitively, the aforementioned results show that as the circuit satisfiability algorithms improve in running time and scope, they imply stronger lower bounds. In all known results, to prove a lower bound against $\mathcal{C}$, one must design a SAT algorithm for a circuit class that is at least as powerful as $\mathcal{C}$. Inspecting the proofs of the above theorems carefully, it is not hard to show that, even if $\mathcal{C}$ did not satisfy the desired closure properties, it would suffice to give a SAT algorithm for a slightly more powerful class than the lower bound. For example, in Theorem 2, a SAT algorithm running in $O(2^{n-n^\varepsilon})$ time for 2^{n^ε}-size AND of ORs of three (possibly negated) $\mathcal{C}$ circuits (on n inputs, of 2^{n^ε} size) would still imply $\mathcal{C}$-circuit lower bounds for Quasi-NP. Our key point here is that *these proof methods require a SAT algorithm for a potentially more powerful circuit class than the class for which we can conclude a lower bound.* A compelling question is whether this requirement is an artifact of our proof method, or is it inherent?

Lower bounds for more powerful classes from SAT algorithms?

We feel it is natural to conjecture that a SAT algorithm for a circuit class $\mathcal{C}$ implies a lower bound against a class that is *more powerful* than $\mathcal{C}$, because checking satisfiability is itself a very powerful ability. Intuitively, a non-trivial SAT algorithm for $\mathcal{C}$ on n-input circuits is computing a *uniform OR* of 2^n $\mathcal{C}$-circuits evaluated on fixed inputs, in $o(2^n)$ time. (Recall

[1] It is not necessary to know precisely what these conditions mean, as we will use different conditions in our paper anyway. The important point is that these conditions hold for most interesting circuit classes that have been studied, such as AC^0, TC^0, NC^1, NC, and general fan-in two circuits.

[2] In this paper, we use the notation $\text{Quasi-NP} := \bigcup_k \mathrm{NTIME}[n^{(\log n)^k}]$.

that a "uniform" circuit informally means that any gate of the circuit can be efficiently computed by an algorithm.) If there were an algorithm to decide the outputs of uniform ORs of $\mathcal{C}$-circuits more efficiently than their actual circuit size, perhaps this implies a lower bound against $\mathrm{OR} \circ \mathcal{C}$ circuits.

Similarly, a #SAT algorithm for $\mathcal{C}$ on n-input circuits can be used to compute the output of any circuit of the form $f(C(x_1), \ldots, C(x_{2^n}))$ where f is a uniform symmetric Boolean function, C is a $\mathcal{C}$-circuit with n inputs, and $x_1, \ldots, x_{2^n}$ is an enumeration of all n-bit strings. Should we therefore expect to prove lower bounds on symmetric functions of $\mathcal{C}$-circuits, using a #SAT algorithm? This question is particularly significant because in many of the concrete lower bounds proved via the program [25, 21, 16], non-trivial #SAT algorithms were actually obtained, not just SAT algorithms. So our question amounts to asking: *how strong of a circuit lower bound we can prove, given the SAT algorithms we already have?* We use SYM to denote the class of Boolean symmetric functions.

▶ **Conjecture 1** (#SAT Algorithms Imply Symmetric Function Lower Bounds, Informal)**.** *Non-trivial #SAT algorithms for circuit classes $\mathcal{C}$ imply size lower bounds against $\mathsf{SYM} \circ \mathcal{C}$ circuits. In particular, all statements in Theorem 1 and Theorem 2 hold when the SAT algorithm is replaced by a #SAT algorithm, and the lower bound consquence for $\mathcal{C}$ is replaced by $\mathsf{SYM} \circ \mathcal{C}$.*

If Conjecture 1 is true, then existing #SAT algorithms would already imply super-polynomial lower bounds for $\mathsf{SYM} \circ \mathsf{ACC}^0 \circ \mathrm{THR}$ circuits, a class that contains depth-two symmetric circuits (for which no lower bounds greater than n^2 are presently known) [18, 1].

More intuition for Conjecture 1 can be seen from a recent paper of the second author, who showed how #SAT algorithms for a circuit class $\mathcal{C}$ can imply lower bounds on *(real-valued) linear combinations of $\mathcal{C}$-circuits* [23]. For example, known #SAT algorithms for ACC^0 circuits imply $\mathsf{Quasi\text{-}NP}$ problems cannot be computed via polynomial-size linear combinations of polynomial-size $\mathsf{ACC}^0 \circ \mathrm{THR}$ circuits. However, the linear combination representation is rather constrained: the linear combination is required to always output 0 or 1. Applying PCPs of proximity, Chen and Williams [6] showed that the lower bound of [23] can be extended to "approximate" linear combinations of $\mathcal{C}$-circuits, where the linear combination does not have to be exactly 0 or 1, but must be closer to the correct value than to the incorrect one, within an additive constant factor. These results show, in principle, how a #SAT algorithm for a circuit class $\mathcal{C}$ can imply lower bounds for a stronger class of representations than $\mathcal{C}$.

1.1 Conjecture 1 Holds for Sparse Symmetric Functions

In this paper, we take a concrete step towards realizing Conjecture 1, by proving it for "sparse" symmetric functions. We say a symmetric Boolean function $f(x_1, \ldots, x_n)$ is *k-sparse* if f is 1 on at most k values of $\sum_i x_i$. The 1-sparse symmetric functions are called the *exact threshold* (ETHR with polynomial weights) or *exact majority* (EMAJ) functions, which have been studied for years in both circuit complexity (e.g. [11, 4, 12, 13, 14]) and structural complexity theory, where the corresponding complexity class (computing an exact majority over all computation paths) is known as $\mathsf{C_{=}P}$ [20].

▶ **Theorem 3.** *Let $\mathcal{C}$ be closed under AND_2, negation, and suppose the all-ones and parity function are in $\mathcal{C}$. Let $f = \{f_n\}$ be a family of k-sparse symmetric functions for some $k = O(1)$.*

- *If there is a #SAT algorithm for n^k-size $\mathcal{C}$-circuits running in $2^n/n^k$ time (for all k), then NEXP does not have $f \circ \mathcal{C}$-circuits of polynomial size.*
- *If there is a #SAT algorithm for 2^{n^ε}-size $\mathcal{C}$-circuits running in 2^{n-n^ε} time (for some $\varepsilon > 0$), then $\mathsf{Quasi\text{-}NP}$ does not have $f \circ \mathcal{C}$-circuits of polynomial size.*

Applying known #SAT algorithms for $\mathsf{AC}^0[m] \circ \mathrm{THR}$ circuits from [22], we obtain:

▶ **Corollary 4.** *For all constant depths $d \geq 2$ and constant moduli $m \geq 2$,* Quasi-NP *does not have polynomial-size EMAJ $\circ$ $\mathsf{AC}^0[m] \circ$ THR circuits.*

1.2 Intuition

Here we briefly explain the new ideas that lead to our new circuit lower bounds.

As in prior work [23, 6], the high-level idea is to show that if (for example) Quasi-NP has polynomial-size EMAJ $\circ \mathcal{C}$, and there is a #SAT algorithm for $\mathcal{C}$ circuits, then we can design a nondeterministic algorithm for verifying GAP Circuit Unsatisfiability (GAP-UNSAT) on generic circuits that beats exhaustive search. In GAP-UNSAT, we are given a generic circuit and are promised that it is either unsatisfiable, or at least half of its possible assignments are satisfying, and we need to nondeterministically prove the unsatisfiable case. (Note this is a much weaker problem than SAT.) As shown in [24, 25, 16], combining a nondeterministic algorithm for GAP-UNSAT with the hypothesis that Quasi-NP has polynomial-size circuits, we can derive that nondeterministic time 2^n can be simulated in time $o(2^n)$, contradicting the nondeterministic time hierarchy theorem.

Our key idea is to use probabilistically checkable proofs (PCPs) in a new way to exploit the power of a #SAT algorithm. First, let's observe a task that a #SAT algorithm for $\mathcal{C}$ can compute on an EMAJ $\circ \mathcal{C}$ circuit. Suppose our EMAJ $\circ \mathcal{C}$ circuit has the form

$$D(x) = \left[\sum_{i=1}^{t} C_i(x) = s \right],$$

where each $C_i(x)$ is a Boolean $\mathcal{C}$-circuit on n inputs, s is a threshold value, and our circuit outputs 1 if and only if the sum of the C_i's equals s.[3] Consider the expression

$$E(x) := \left(\sum_{i=1}^{t} C_i(x) - s \right)^2. \tag{1}$$

Treated as a function, $E(x)$ outputs integers; $E(a) = 0$ when $D(a) = 1$, and otherwise $E(a) \in [1, (t+s)^2]$. We first claim that the quantity

$$\sum_{a \in \{0,1\}^n} E(a) \tag{2}$$

can be compute faster than exhaustive search using a faster #SAT algorithm. To see this, using distributivity, we can rewrite (1) as

$$E(x) = \sum_{i,j} (C_i \wedge C_j)(x) - 2s \sum_i C_i(x) + s^2.$$

Assuming $\mathcal{C}$ is closed under conjunction, each $C_i \wedge C_j$ is also a $\mathcal{C}$-circuit, and we can compute

$$\sum_{a \in \{0,1\}^n} E(a) = \sum_{i,j} \left(\sum_{a \in \{0,1\}^n} (C_i \wedge C_j)(a) \right) - 2s \sum_i \left(\sum_{a \in \{0,1\}^n} C_i(a) \right) + s^2 \cdot 2^n$$

by making $O(t^2)$ calls to a #SAT algorithm. Thus we can compute (2) using a #SAT algorithm.

[3] We are using the standard Iverson bracket notation, where $[P]$ is 1 if predicate P is true, and 0 otherwise.

How is computing (2) useful? This is where PCPs come in. We cannot use (2) to directly solve #SAT for D (otherwise as #SAT algorithms imply SAT algorithms we could apply existing work [25], and be done). But we can use (2) to obtain a *multiplicative approximation* to the number of assignments that *falsify* D. In particular, each satisfying assignment is counted zero times in (2), and each falsifying assignment is counted between 1 and (less than) $(t+s)^2$ times. We want to exploit this, and obtain a faster GAP-UNSAT algorithm. Given a circuit which is a GAP-UNSAT instance, we start by using an efficient hitting set construction [10] to increase the gap of GAP-UNSAT. We obtain a new circuit $C(x)$ which is either UNSAT or has at least $2^n - o(2^n)$ satisfying assignments (Section 2.1). Next (Lemma 15) we apply a PCP of Proximity and an error correcting code to C, yielding a 3-SAT instance over x and extra variables, with constant gap (similar to Chen-Williams [6]), and we amplify this gap using standard serial repetition. Finally, we apply the FGLSS [9] reduction (Lemma 19) to the 3-SAT instance, obtaining Independent Set instances with a large gap between the YES case and NO case. In particular, for all inputs x, when $C(x) = 1$ there is a large independent set in the resulting graph, and when $C(x) = 0$, there are only small independent sets in the resulting graph (see Lemma 14). Returning to the assumption that Quasi-NP has small EMAJ $\circ\, \mathcal{C}$ circuits, and applying an easy witness lemma [16], it follows that the solutions to the independent set instance can be encoded by EMAJ $\circ\, \mathcal{C}$ circuits. Because of the large gap between the YES case and NO case, our multiplicative approximation to the number of UNSAT assignments can be used to distinguish the unsatisfiable case and the "many satisfying assignments" case of GAP-UNSAT, which finishes the argument.

One interesting bottleneck is that we cannot *directly* apply serial repetition and the FGLSS reduction in our argument; we need the PCP machinery we use to behave similarly on all inputs x to the original circuit C. This translates to studying the behavior of these reductions *with respect to partial assignments*. While for these two reductions we are able to prove that they behave "nicely" with respect to partial assignments, it is entirely unclear that this is true for other PCP reductions such alphabet reduction, parallel repetition, and so on.

Our approach is very general; to handle k-sparse symmetric functions, we can simply modify the function E accordingly.

2 Preliminaries and Organization

We assume general familiarity with basic concepts in circuit complexity and computational complexity [2]. In particular we assume familiarity with AC^0, ACC^0, $\mathsf{P}_{/\mathsf{poly}}$, NEXP, and so on.

Circuit Notation

Here we define notation for the relevant circuit classes. By $\text{size}_{\mathcal{C}}(h(n))$ we denote circuits from circuit class $\mathcal{C}$ with size at most $h(n)$.

▶ **Definition 5.** *An EMAJ$\circ\mathcal{C}$ circuit (a.k.a. "exact majority of $\mathcal{C}$ circuit") has the general form $EMAJ(C_1(x), C_2(x), \ldots, C_t(x), u)$, where u is a positive integer, x are the input variables, $C_i \in \mathcal{C}$, and the gate $EMAJ(y_1, \ldots, y_t, u)$ outputs 1 if and only if exactly u of the y_i's output 1.*

▶ **Definition 6.** *A $SUM^{\geq 0} \circ \mathcal{C}$ circuit ("positive sum of $\mathcal{C}$ circuits") has the form*

$$SUM^{\geq 0}(C_1(x), C_2(x), \ldots, C_t(x)) = \sum_{i \in [t]} C_i(x)$$

where C_i is either a $\mathcal{C}$-circuit or -1 times a $\mathcal{C}$-circuit and we are promised that $\sum_{i\in[t]} C_i(x) \geq 0$ over all $x \in \{0,1\}^n$.

Given a set of circuits $\{C_i\}$, we say that $f : \{0,1\}^n \to \{0,1\}$ is represented *by the positive-sum circuit $SUM^{\geq 0}(C_1(x), C_2(x), \ldots, C_t(x))$ if for all x, $f(x) = 1$ when $\sum_{i\in[t]} C_i(x) > 0$, and $f(x) = 0$ when $\sum_{i\in[t]} C_i(x) = 0$.*

▶ **Definition 7.** *A circuit class $\mathcal{C}$ is* typical *if there is a $k > 0$ such that the following hold:*

- ***Closure under negation.*** *For every $\mathcal{C}$ circuit C, there is a circuit C' computing the negation of C where $size(C') \leq size(C)^k$.*
- ***Closure under AND.*** *For every $\mathcal{C}$ circuits C_1 and C_2, there is a circuit C' computing the AND of C_1 and C_2 where $size(C') \leq (size(C_1) + size(C_2))^k$.*
- ***Contains all-ones.*** *The function $\mathbf{1}_n : \{0,1\}^n \to \{0,1\}$ has a $\mathcal{C}$ circuit of size $O(n^k)$.*

The vast majority of circuit classes that are studied (AC^0, ACC^0, TC^0, NC^1, $\mathsf{P}_{/\mathsf{poly}}$) are typical.[4] The next lemma shows that the negation of an exact-majority of $\mathcal{C}$ circuit can be represented as a "positive-sum" of $\mathcal{C}$ circuit, if $\mathcal{C}$ is typical.

▶ **Lemma 8.** *Let $\mathcal{C}$ be typical. If a function f has a $EMAJ \circ \mathcal{C}$ circuit D of size s, then $\neg f$ can be represented by a $SUM^{\geq 0} \circ \mathcal{C}$ circuit D' of size $poly(s)$. Moreover, a description of the circuit D' can be obtained from a description of D in polynomial time.*

Proof. Suppose f is computable by the $\mathrm{EMAJ} \circ \mathcal{C}$ circuit $D = \mathrm{EMAJ}(D_1, D_2, \ldots, D_t, u)$, where $u \in \{0, 1, \ldots, t\}$. Consider the expression

$$E(x) := (\mathrm{SUM}(D_1, D_2, \ldots, D_t) - u)^2.$$

Note that $E(x) = 0$ when $D(x) = 1$, and $E(x) > 0$ when $D(x) = 0$. So in order to prove the lemma, it suffices to show that E can be written as a $\mathrm{SUM}^{\geq 0} \circ \mathcal{C}$ circuit. Expanding the expression E,

$$\begin{aligned} E(x) &= \mathrm{SUM}(D_1, D_2, \ldots, D_t)^2 - 2u \cdot \mathrm{SUM}(D_1, D_2, \ldots, D_t) + u^2 \\ &= \sum_{i,j=1}^{t} (D_i \wedge D_j) - \sum_{j=1}^{2u} \sum_{i=1}^{t} D_i + u^2. \end{aligned}$$

By Definition 7 $\mathrm{AND}_2 \circ \mathcal{C} = \mathcal{C}$, each $D_i \wedge D_j$ is a circuit from $\mathcal{C}$ of size $\mathrm{poly}(s)$. Since the all-ones function is in $\mathcal{C}$, the function $x \mapsto u^2$ also has a $\mathrm{SUM} \circ \mathcal{C}$ circuit of size $O(t^2)$. Therefore there are circuits $D'_i \in \mathcal{C}$ and $t' \leq O(t^2)$ such that by defining $D' := \mathrm{SUM}^{\geq 0}(D'_1, \ldots, D'_{t'})$ we have $D'(x) = E(x)$ for all x. ◀

Error-Correcting Codes

We will need a (standard) construction of binary error correcting codes with constant rate and constant relative distance.

▶ **Theorem 9** ([17])**.** *There are universal constants $c \geq 1$ and $\delta \in (0,1)$ such that for all sufficiently large n, there are linear functions $ENC^n : (\mathbb{F}_2)^n \to (\mathbb{F}_2)^{cn}$ such that for all $x \neq y$ with $|x| = |y| = n$, the Hamming distance between $ENC^n(x)$ and $ENC^n(y)$ is at least δn.*

In what follows, we generally drop the superscript n for notational brevity. Note that each bit of output $\mathrm{ENC}^n_i(x)$ (for $i = 1, \ldots, cn$) is a parity function on some subset of the input bits.

[4] A notable exception (as far as we know) is the class of depth-d *exact* threshold circuits for a fixed $d \geq 2$, because we do not know if such classes are closed under negation. Similarly, we do not know if the class of depth-d threshold circuits is typical. (In that case, the only non-trivial property to check is closure under AND; we can compute the AND of two threshold circuits with a quasi-polynomial blowup using Beigel-Reingold-Spielman [3], but not with a polynomial blowup.)

2.1 Weak CAPP Algorithms Are Sufficient For Lower Bounds

Murray and Williams [16] showed that CAPP/GAP-UNSAT algorithms, i.e., algorithms which distinguish between unsatisfiable circuits and circuits with $\geq 2^{n-1}$ satisfying assignments are enough to give lower bounds. For our results, it is necessary to strengthen the "gap", which can be done using known hitting set constructions.

▶ **Lemma 10** (Corollary C.5 in [10], Hitting Set Construction). *There is a constant $\psi > 0$ and a $\text{poly}(n, \log g)$ time algorithm S such that, given a (uniform random) string r of $n+\psi\cdot\log g$ bits, S outputs $t = O(\log g)$ strings $x_1, x_2, \ldots, x_t \in \{0,1\}^n$ such that for every $f : \{0,1\}^n \to \{0,1\}$ with $\sum_x f(x) \geq 2^{n-1}$, $\Pr_r[OR_{i=1}^t f(x_i) = 1] \geq 1 - 1/g$.*

We will use the following "algorithms to lower bounds" connections as black box:

▶ **Theorem 11** ([16]). *Suppose for some constant $\varepsilon \in (0,1)$ there is an algorithm A that for all 2^{n^ε}-size circuits C on n inputs, $A(C)$ runs in 2^{n-n^ε} time, outputs YES on all unsatisfiable C, and outputs NO on all C that have at least 2^{n-1} satisfying assignments. Then for all k, there is a $c \geq 1$ such that $NTIME[2^{\log^{ck^4/\varepsilon} n}] \not\subset \mathsf{SIZE}[2^{\log^k n}]$.*

Applying Lemma 10 to Theorem 11, we observe that the circuit lower bound consequence can be obtained from a significantly weaker-looking hypothesis. This weaker hypothesis will be useful for our lower bound results.

▶ **Theorem 12.** *Suppose for some constant $\varepsilon \in (0,1)$ there is an algorithm A that for all 2^{n^ε}-size circuits C on n inputs, $A(C)$ runs in $2^n/g(n)^{\omega(1)}$ time, outputs YES on all unsatisfiable C, and outputs NO on all C that have at least $2^n(1-1/g(n))$ satisfying assignments, for $g(n) = 2^{n^{2\varepsilon}}$. Then for all k, there is a $c \geq 1$ such that $NTIME[2^{\log^{ck^4/\varepsilon} n}] \not\subset \mathsf{SIZE}[2^{\log^k n}]$.*

Proof. Our starting point is Theorem 11 ([16]): we are given an m-input, 2^{m^δ}-size circuit D' that is either UNSAT or has at least 2^{m-1} satisfying assignments, and we wish to distinguish between the two cases with a 2^{m-m^δ}-time algorithm. We set $\delta = \varepsilon/2$

We create a new circuit D with n inputs, where n satisfies

$$n = m + \psi \cdot \log g(n),$$

and $\psi > 0$ is the constant from Lemma 10. (Note that, since $g(n)$ is time constructible and $g(n) \leq 2^{o(n)}$, such an n can be found in subexponential time.) Applying the algorithm from Lemma 10, D treats its n bits of input as a string of randomness r, computes $t = O(\log g(n))$ strings $x_1, x_2, \ldots, x_t \in \{0,1\}^m$ with a $\text{poly}(m, \log g)$-size circuit, then outputs the OR of $D'(x_i)$ over all $i = 1, \ldots, t$. Note the total size of our circuit D is $\text{poly}(m, \log g) + O(\log g) \cdot \text{size}(D') = \text{poly}(n) + O(n^{2\varepsilon}) \cdot 2^{m^\delta} < 2^{n^{2\delta}} = 2^{n^\varepsilon}$ as $\varepsilon = 2\delta$.

Clearly, if D' is unsatisfiable, then D is also unsatisfiable. By Lemma 10, if D' has 2^{m-1} satisfying assignments, then D has at least $2^n(1-1/g(n))$ satisfying assignments. As $\text{size}(D) \leq 2^{n^\varepsilon}$, by our assumption we can distinguish the case where D is unsatisfiable from the case where D has at least $2^n(1-1/g(n))$ satisfying assignments, with an algorithm running in time $2^n/g(n)^{\omega(1)}$. This yields an algorithm for distinguishing the original circuit D' on m inputs and 2^{m^δ} size, running in time

$$2^n/g(n)^{\omega(1)} = 2^m g(n)^{O(1)}/g(n)^{\omega(1)} = 2^m/g(n)^{\omega(1)} \leq 2^m 2^{-n^{2\varepsilon}} \leq 2^m 2^{-n^\delta} \leq 2^{m-m^\delta},$$

since $g(n) = 2^{n^{2\varepsilon}}$. By Theorem 11, this implies that for all k, there is a $c \geq 1$ such that $\text{NTIME}[2^{\log^{ck^4/\delta} n}] \not\subset \mathsf{SIZE}[2^{\log^k n}]$. As, $\varepsilon = 2\delta$ we get that $\text{NTIME}[2^{\log^{2ck^4/\varepsilon} n}] \not\subset \mathsf{SIZE}[2^{\log^k n}]$. But as the constant 4 can be absorbed in the constant c hence we get that for all k, there is a $c \geq 1$ such that $\text{NTIME}[2^{\log^{ck^4/\varepsilon} n}] \not\subset \mathsf{SIZE}[2^{\log^k n}]$. ◀

2.2 Organization

In Section 3 we give a reduction from Circuit SAT to "Generalized" Independent Set. Section 4 uses this reduction to prove lower bounds for EMAJ$\circ\mathcal{C}$ assuming #SAT algorithms for $\mathcal{C}$ with running time $2^{n-n^{\varepsilon}}$. Section 4.1 uses this result to give lower bound for EMAJ$\circ$ACC$^0\circ$THR. Section 5 generalizes these results to $f\circ\mathcal{C}$ lower bounds where f is a sparse symmetric function. In the full version of the paper [19] we give lower bounds for EMAJ$\circ\mathcal{C}$ assuming #SAT algorithms for $\mathcal{C}$ with running time $2^n/n^{\omega(1)}$.

3 From Circuit SAT to Independent Set

The goal of this section is to give the main PCP reduction we will use in our new algorithm-to-lower-bound theorem. First we need a definition of "generalized" independent set instances, where some vertices have already been "assigned" in or out of the independent set.

▶ **Definition 13.** *Let $G=(V,E)$ be a graph. Let $\pi: V\to\{0,1,*\}$ be a partial Boolean assignment to V. We define $G(\pi)$ to be a graph with the label function π on its vertices (where each vertex gets the label 0, or 1, or no label). We construe $G(\pi)$ as an* ***generalized independent set instance****, in which any valid independent set (vertex assignment) must be consistent with π: any independent set must contain all vertices labeled 1, and no vertices labeled 0.*

▶ **Lemma 14.** *Let k be a function of n. Given a circuit D on X with $|X|=n$ bits and of size $m>n$, there is a poly$(m,2^{O(k)})$-time reduction from D to a generalized independent set instance on graph $G_D=(V_D,E_D)$, with the following properties.*

- *Each vertex $v\in V_D$ is associated with a set of pairs S_v of the form $\{(i,b)\}\subseteq[O(n)]\times\{0,1\}$. The set $\{S_v\}$ is produced as part of the reduction.*
- *Each assignment x to X defines a partial assignment π_x to V_D such that*

$$\pi_x(v)=\begin{cases}0 & \text{if } \exists(i,b)\in S_v \text{ such that } ENC_i(x)\neq b\\ * & \text{otherwise,}\end{cases}$$

  where ENC is the error-correcting code from Theorem 9.
- *If $D(x)=0$, the maximum independent set in $G_D(\pi_x)$ equals κ for an integer κ, and furthermore given x, it can be found in time poly$(n,m,2^{O(k)})$.*
- *If $D(x)=1$, then the maximum independent set in $G_D(\pi_x)$ has size at most $\kappa/2^k$.*

Intuitively, the use of Lemma 14 is that we will start with a "no satisfying assignment" vs "most assignments are satisfying" GAP-UNSAT instance from Theorem 12. Now in the "no satisfying assignment" case for all x the reduced independent set instance $G_D(\pi_x)$ has a large independent set instance. Counting the sum of independent sets over x gives a high value. On the other hand in the "most assignments are satisfying" case for most x the reduced independent set instance $G_D(\pi_x)$ has a small independent set and for a very few x, $G_D(\pi_x)$ can have a large independent set. Hence in this case counting the sum of independent sets over all x gives a low value. The difference between the high value and low value is big enough that even a approximate counting of these values as outlined in Section 1.2 is enough to distinguish and hence solve the GAP-UNSAT instance.

The remainder of this section is devoted to the proof of Lemma 14.

Let us set up some notation for variable assignments to a formula. Let F be a SAT instance on a variable set Z, and let $\tau: Z\to\{0,1,\star\}$ be a partial assignment to Z. Then we define $F(\tau)$ to be the formula obtained by setting the variables in F according to τ. Note that we do not perform further reduction rules on the clauses in $F(\tau)$: for each clause in F that becomes false (or true) under τ, there is a clause in $F(\tau)$ which is always false (true).

For every subsequence Y of variables from Z, and every vector $y \in \{0,1\}^{|Y|}$, we define $F(Y = y)$ to be the formula F in which the i^{th} variable in Y is assigned y_i, and all other variables are left unassigned.

▶ **Lemma 15** (PCPP+ECC, [6]). *There is a polynomial-time transformation that, given a circuit D on n inputs of size $m \geq n$, outputs a 3-SAT instance F on the variable set $Y \cup Z$, where $|Y| \leq poly(n)$, $|Z| \leq poly(m)$, and the following hold for all $x \in \{0,1\}^n$:*

- *If $D(x) = 0$ then $F(Y = ENC(x))$ on variable set Z has a satisfying assignment z_x. Furthermore, there is a $poly(m)$-time algorithm that given x outputs z_x.*
- *if $D(x) = 1$ then there is no assignment to the Z variables in $F(Y = ENC(x))$ satisfying more than a $(1 - \Omega(1))$-fraction of the clauses.*

where $ENC : \{0,1\}^n \to \{0,1\}^{O(n)}$ is the linear encoding function from Theorem 9. As it is a linear function, the i^{th} bit of output $ENC_i(x)$ satisfies $ENC_i(x) = \oplus_{j \in U_i} x_j$ for some set U_i.

Serial Repetition [8] is a basic operation on CSPs/PCPs, in which a new CSP is created whose constraints are ANDs of k uniformly sampled clauses from the original CSP. Serial repetition is usually done for the purpose of reducing soundness, i.e., reducing the fraction of satisfiable clauses. We now state a derandomized version of serial repetition.

▶ **Lemma 16** (Serial repetition [8]). *Given a 3-SAT instance F on n variables denoted by Y with m clauses we can construct a $O(k)$-SAT formula F' on the same n variables with $m2^{O(k)}$ clauses such that:*

1. *If $Y = y$ satisfies F then y satisfies F'.*
2. *If $F(Y = y)$ is at most $1 - \Omega(1)$ satisfiable then $F'(Y = y)$ is at most $1/2^k$ satisfiable.*

Next we prove a stronger version of derandomized serial repetition with guarantees for partial assignments. The proof directly follows from the guarantees of standard Serial Repetition (Lemma 16).

▶ **Lemma 17** (Serial repetition with partial assignments). *Let k be a function of n. Given a 3-SAT instance F on n variables denoted by Y, Z with m clauses we can construct a $O(k)$-SAT formula F' on the same n variables with $m \cdot 2^{O(k)}$ clauses such that:*

1. *If $Y, Z = y, z$ satisfies F then y, z satisfies F'.*
2. *If $F(Y = y)$ is at most $1 - \Omega(1)$ satisfiable then $F'(Y = y)$ is at most $1/2^k$ satisfiable.*

Proof. We prove that just standard serial repetition from Lemma 16 suffices for proving this stronger property.

Property 1 directly follows from Property 1 in Lemma 16.

Define $F_y = F(Y = y)$ where we treat any clauses that became FALSE or TRUE under $Y = y$ as normal clauses. Let F'_y be the $O(k)$-SAT formula obtained by applying serial repetition to f_y from Lemma 17.

In Serial Repetition [8] it is clear that clauses in F' are just ANDs of clauses in F and which clauses are part of the "AND" is only dependent on their index.

Due to this $F'(Y = y)$ i.e. first applying serial repetition then setting $Y = y$ is equivalent to first setting $Y = y$ and then applying serial repetition i.e. F'_y.

By our assumption F_y is at most $1 - \Omega(1)$ satisfiable and hence by Property 2 of Lemma 16 F'_y is at most $1/2^k$ satisfiable. As $F'_y = F'(Y = y)$ we have that $F'(Y = y)$ is at most $1/2^k$ satisfiable. ◀

The FGLSS reduction [9] maps a CSP Φ to a graph G_Φ such that the MAX-SAT value in Φ is equal to the size of the maximum independent set in G_Φ.

▶ **Lemma 18** (FGLSS [9]). *Let F be a k-SAT instance on variable set Y with $|Y| = n$ and m clauses. There exists a poly$(n, m, 2^{O(k)})$ time reduction graph from F to a graph $G_F = (V_F, E_F)$ such that: the size of maximum independent set in G_F is exactly equal to maximum clauses satisfiable in F.*

We note that a stronger version of the FGLSS reduction [9] holds with guarantees for partial assignments. The proof is very similar to the proof of the standard FGLSS reduction (Lemma 18).

▶ **Lemma 19** (FGLSS with partial assignments). *Let F be a k-SAT instance on variable set Y, Z with $|Y| + |Z| = n$ and m clauses. There exists a poly$(n, m, 2^{O(k)})$ time reduction graph from F to an independent set instance on graph $G_F = (V_F, E_F)$. Each vertex $v \in V_F$ is a associated to a set T_v of $(i \in [|Y|], b \in \{0, 1\})$ pairs. For each partial assignment of the form $\tau : Y \to \{0, 1\}$ define a partial assignment π_τ to V_F such that:*

$$\pi_\tau(v) = \begin{cases} 0 & \text{if } \exists (i, b) \in T_v \text{ such that } \tau(Y_i) \neq b \\ * & \text{otherwise,} \end{cases}$$

Then the max independent set in $G_F(\pi_\tau)$ equals the max number of clauses satisfiable in $F(\tau)$.

Proof. Let w be a clause in F and w_i denote the i^{th} variable in w. Let ℓ denote a satisfying assignment to w. For every w, ℓ pair create a vertex in V_F. Let v be the vertex associated with a particular w, ℓ. Let $T_v = \{(w_i, \ell_i\}$ represent the assignment $w_i = \ell_i$ for $1 \leq i \leq k$.

Make an edge between vertex u and vertex v if the assignment T_u and T_v contradict each other. Note that this means that there is always an edge between two vertices associated to the same clause but different satisfying assignments i.e. vertices associated with the same clause form a clique.

Let x be a assignment for F satisfying κ clauses. We now give an independent set in G_F of size κ. For every satisfied clause w and and ℓ the assignment to variables of w in x we choose the vertex w, ℓ in the independent set. As there are κ satisfied clauses we choose κ vertices. These vertices form and independent set as if two of these vertices u, v had an edge between them it would mean that the assignments T_u and T_v contradict each other. This is not possible as all these assignments are partial assignments of x.

Consider S to be an independent set in G_F of size κ. We now give an assignment to F which satisfies κ clauses. Note that from vertices corresponding to the same clauses only 1 vertex can be a part of independent set as they all form a clique. Hence vertices associated with κ different clauses must be part of the independent set. For a vertex u associated with w, ℓ the partial assignment T_u satisfies w. For two vertices u, v in the independent set the partial assignments from T_v and T_u do not contradict as otherwise there would be an edge between u and v. Hence we can join all the partial assignments T_v for vertices v in the independent set to get a partial assignment which satisfies κ clauses in $F(\tau)$. Hence the maximum independent set in $G_F(\pi_\tau)$ has size at most the maximum number clauses satisfied in $F(\tau)$. ◀

We next present the proof of Lemma 14 which just follows by combining Lemma 15, 17, and 19 sequentially.

Proof of Lemma 14. The proof follows by applying Lemma 15, 17 and 19 sequentially.

We start from a circuit D with input variables X ($|X| = n$) and size $m > n$. Lemma 15 transform this into a 3-SAT instance F with poly(m) clauses on the variable set $Y \cup Z$, where $|Y| \leq \text{poly}(n)$, $|Z| \leq \text{poly}(m)$, and the following hold for all $x \in \{0, 1\}^n$:

- If $D(x) = 0$ then $F(Y = \mathrm{ENC}(x))$ on variable set Z has a satisfying assignment z_x. Furthermore, there is a $\mathrm{poly}(m)$-time algorithm that given x outputs z_x.
- if $D(x) = 1$ then there is no assignment to the Z variables in $F(Y = \mathrm{ENC}(x))$ satisfying more than a $(1 - \Omega(1))$-fraction of the clauses.

where $\mathrm{ENC} : \{0,1\}^n \to \{0,1\}^{O(n)}$ is the linear encoding function from Theorem 9.

Applying Lemma 17 on F gives us a $O(k)$-SAT formula F' on the same $Y \cup Z$ variables with $\mathrm{poly}(m) \cdot 2^{O(k)}$ clauses such that:

1. If $Y, Z = y, z$ satisfies F then y, z satisfies F'.
2. If $F(Y = y)$ is at most $1 - \Omega(1)$ satisfiable then $F'(Y = y)$ is at most $1/2^k$ satisfiable.

which implies that:

- If $D(x) = 0$ then $F'(Y = \mathrm{ENC}(x))$ on variable set Z has a satisfying assignment z_x. Furthermore, there is a $\mathrm{poly}(m)$-time algorithm that given x outputs z_x.
- if $D(x) = 1$ then there is no assignment to the Z variables in $F'(Y = \mathrm{ENC}(x))$ satisfying more than a $1/2^k$-fraction of the clauses.

Finally applying Lemma 19 to F' where we consider partial assignments τ which assign Y to $\mathrm{ENC}(x)$ for some x. Hence $\tau(Y_i) = \mathrm{ENC}_i(x)$. As τ is fixed by fixing x we rename π_τ to π_x. S_v is just a renaming of T_v. Size of the graph is $\mathrm{poly}(n + m, \mathrm{poly}(m) \cdot 2^{O(k)}, 2^{O(k)}) = \mathrm{poly}(m, 2^k)$ as $m > n$. ◀

4 Main Result

We now turn to the proof of the main result, Theorem 3. We will prove the result for $\mathrm{EMAJ} \circ \mathcal{C}$ first, and sketch how to extend to $f \circ \mathcal{C}$ for sparse symmetric f in Section 5. Below we prove $\mathrm{EMAJ} \circ \mathcal{C}$ lower bounds for Quasi-NP when we have 2^{n-n^ε} time algorithms for #SAT on $\mathcal{C}$ circuits of size 2^{n^ε}. For the other parts of Theorem 3 (on #SAT algorithms with running time $2^n/n^{\omega(1)}$), see the full version of the paper [19].

We note here that in Theorem 3 we mentioned polynomial size lower bounds for $\mathrm{EMAJ} \circ \mathcal{C}$ we in fact prove quasi-polynomial size lower bounds below.

▶ **Theorem 20.** *Suppose $\mathcal{C}$ is typical, and the parity function has poly(n)-sized $\mathcal{C}$ circuits. Then for every k,* quasi-NP *does not have $\mathrm{EMAJ} \circ \mathcal{C} = \mathcal{H}$ circuits of size $O(n^{\log^k n})$, if for some $\epsilon \in (0,1)$ there is a #SAT algorithm running in time 2^{n-n^ε} for all circuits from class $\mathcal{C}$ of size at most 2^{n^ε}.*

Proof. Let us assume that for a fixed $k > 0$, quasi-NP has $\mathcal{H} = \mathrm{EMAJ} \circ \mathcal{C}$ circuits of size $O(n^{(\log^k n)})$ which implies that quasi-NP $\in \mathrm{size}(n^{O(\log^k n)})$ for general circuits. By Theorem 12, we obtain a contradiction if for some constant $\delta \in (0,1)$ and $g(n) = 2^{n^{2\delta}}$ we can give a $2^n/g(n)^{\omega(1)}$ time nondeterministic algorithm for distinguishing between:

1. YES case: D has no satisfying assignments.
2. NO case: D has at least $2^n\,(1 - 1/g(n))$ satisfying assignments

given a generic fan-in 2 circuit D with n inputs and size $m \leq h(n) := 2^{n^\delta}$. Under the hypothesis, we will give such an algorithm for $\delta = \varepsilon/4$.

Using Lemma 14, we reduce the circuit D to an independent set instance G_D (with $k = \log h(n)$) on $n_2 = \mathrm{poly}(m, 2^{O(k)}) = \mathrm{poly}(m, 2^{O(k)}) = \mathrm{poly}(m, h(n)^{O(1)}) = \mathrm{poly}(h(n))$ vertices. We also find subsets S_i for every vertex $i \in [n_2]$. Let π_x be the partial assignment which assigns a vertex i to 0 if there exist $(j', b) \in S_i$ such that $\mathrm{ENC}_{j'}(x) \neq b$. Note that π_x does not assign any vertex to 1. By Lemma 14, G_D has the following properties:

1. If $D(x) = 0$, then $G_D(\pi_x)$ has an independent set of size κ. Furthermore, given x we can find this independent set in $\text{poly}(h(n))$ time.
2. If $D_1(x) = 1$, then in $G_D(\pi_x)$, all independent sets have size at most $\kappa/h(n)$.

This means it suffices for us to distinguish between the following two cases:
1. YES case: For all x, $G_D(\pi_x)$ has an independent set of size κ.
2. NO case: For at most $2^n/g(n)$ values of $x, G_D(\pi_x)$ has an independent set of size $\geq \kappa/h(n)$.

Guessing a succinct witness circuit: As guaranteed by Lemma 14 given an x such that $D(x) = 0$ we can find the assignment $A(x)$ to G_D which is consistent with π_x and represents an independent set of size κ in $\text{poly}(h(n))$ time. Let $A(x, i)$ denote the assignment to the i^{th} vertex in $A(x)$. Given x and vertex $i \in [n_2]$, in time $\text{poly}(h(n))$ we can produce $\neg A(x, i)$.

▷ Claim 21. Under the hypothesis, there is a $h(n)^{o(1)}$-sized EMAJ $\circ\, \mathcal{C}$ circuit U of size $h(n)^{o(1)}$ with x, i as input representing $\neg A(x, i)$.

Proof. Under the hypothesis, for some constant k, we have quasi-NP $\subseteq \text{size}_{\mathcal{H}}[n^{\log^k n}]$. Specifically, for $p(n) = n^{\log^{k+1} n}$ we have $\text{NTIME}[p(n)] \subseteq \text{size}_{\mathcal{H}}[p(n)^{1/\log n}] \subseteq \text{size}_{\mathcal{H}}[p(n)^{o(1)}]$. As $h(n) = 2^{n^\varepsilon} \gg p(n)$, a standard padding argument implies $\text{NTIME}[\text{poly}(h(n))] \subseteq \text{size}_{\mathcal{H}}[(\text{poly}(h(n)))^{o(1)}] = \text{size}_{\mathcal{H}}[h(n)^{o(1)}]$. Since $\neg A(x, i)$ is computable in $\text{poly}(h(n))$ time, we have that $\neg A(x, i)$ can be represented by a $h(n)^{o(1)}$-sized $\mathcal{H} = \text{EMAJ} \circ \mathcal{C}$ circuit. ◁

Our nondeterministic algorithm for GAP-UNSAT begins by guessing U guaranteed by Claim 21 which is supposed to represent $\neg A$. Then by the reduction in Lemma 8 we can covert U to a $\text{SUM}^{\geq 0} \circ \mathcal{C}$ circuit R for $A(x, i)$ of size $\text{poly}(h(n)^{o(1)}) = h(n)^{o(1)}$. Note that if our guess for U is correct, i.e., $U = \neg A$, then R represents A.

Let the subcircuits of R be $R_1, R_2, \ldots, R_t$, so that $R(x) = \sum_{j\in[t]} R_j$, where $R_j \in \mathcal{C}$ and $t \leq h(n)^{o(1)}$. The number of inputs to R_j is $n' = |x| + \log n_2 = n + O(\log h(n))$, and the size of R_j is $h(n)^{o(1)}$.

Note that $R(x, i) = 0$ represents that the i^{th} vertex is not in the independent set of G_D in a solution corresponding to x, while $R(x, i) > 0$ represents that it is in the independent set of G_D in a solution corresponding to x. For all x and i we have $0 \leq R(x, i) \leq t \leq h(n)^{o(1)}$.

Verifying that R encodes valid independent sets: We can verify that the circuit R produces an independent set on all x by checking each edge over all x. To check the edge between vertices i_1 and i_2 we need to verify that at most one of them is in the independent set. Equivalently, for all x we check that $R(x, i_1) \cdot R(x, i_2) = 0$. As $R(x, i) \geq 0$ for all x and i we can just verify

$$\sum_{x\in\{0,1\}^n} R(x, i_1) \cdot R(x, i_2) = 0.$$

Since $R(x, i) = \sum_{j\in[t]} R_j(x, i)$ it suffices to verify that

$$\sum_{x\in\{0,1\}^n} \sum_{j_1,j_2\in[t]} R_{j_1}(x, i_1) \cdot R_{j_2}(x, i_2) = 0.$$

Let $R_{j_1,j_2}(x, i_1, i_2) = R_{j_1}(x, i_1) \cdot R_{j_2}(x, i_2)$. Since $\mathcal{C}$ is closed under AND (upto polynomial factors) R_{j_1,j_2} also has a $\text{poly}(h(n)^{o(1)}) = h(n)^{o(1)}$ sized $\mathcal{C}$ circuit. Exchanging the order of summations is suffices for us to verify

$$\sum_{j_1,j_2\in[t]} \left(\sum_{x\in\{0,1\}^n} R_{j_1,j_2}(x, i_1, i_2) \right) = 0.$$

For fixed i_1, i_2, j_1, j_2 the number of inputs to R_{j_1,j_2} is $|x| = n$ and its size is $h(n)^{o(1)} \leq 2^{n^\varepsilon}$. Hence, for fixed i_1, i_2, j_1, j_2 we can compute $\sum_x R_{j_1,j_2}(x, i_1, i_2)$ using the #SAT algorithm from our assumption, in time 2^{n-n^ε}. Summing over all j_1, j_2 pairs only adds another multiplicative factor of $t^2 = h(n)^{o(1)}$. This allows us to verify that the edge (i_1, i_2) is satisfied by R. Checking all edges of G_D only adds another multiplicative factor of $\text{poly}(h(n))$. Hence the total running time for verifying that R encodes valid independent sets on all x is still $2^{n-n^\varepsilon}\text{poly}(h(n))$.

Verifying consistency of independent set produced by R with π_x: As we care about the sizes of independent sets in $G_D(\pi_x)$ over all x we need to check if the assignment by R is consistent with π_x. As π_x only assigns vertices to 0, we need to verify that all vertices assigned to 0 in π_x are in fact assigned to 0 by the assignment given by $R(x, \cdot)$. From Lemma 14, we know that π_x assigns a vertex i to 0 if for some $(j', b) \in S_i$, $\text{ENC}_{j'}(x) \neq b$. To check this condition we need to verify that $R(x, i) = 0$ if for some $(j', b) \in S_i$, $\text{ENC}_{j'}(x) \neq b$. Equivalently, we cn check $(\text{ENC}_{j'}(x) \oplus b) \cdot R(x, i) = 0$ for all $x, i, (j', b) \in S_i$. Since $(\text{ENC}_j(x) \oplus b)R(x, i) \geq 0$ for all possible inputs we can just check that

$$\sum_{x \in \{0,1\}^n} (\text{ENC}_{j'}(x) \oplus b) \cdot R(x, i) = 0$$

for all $i, (j', b) \in S_i$. As $R(x, i) = \sum_{j \in [t]} R_j(x, i)$ we can equivalently verify that

$$\sum_{x \in \{0,1\}^n} \sum_{j \in [t]} (\text{ENC}_{j'}(x) \oplus b) \cdot R_j(x, i) = 0$$

for all $i, (j', b) \in S_i$. Note that $R_{j'}(x, i)$ has a $h(n)^{o(1)}$ sized $\mathcal{C}$ circuit. By our assumption parity has a poly(n)-sized $\mathcal{C}$-circuit so $(\text{ENC}_j(x) \oplus b)$ also has a poly(n)-sized $\mathcal{C}$ circuit. Hence $(\text{ENC}_j(x) \oplus b) \cdot R_{j'}(x, i)$ has a $\text{poly}(n, h(n)^{o(1)}) = h(n)^{o(1)}$-sized $\mathcal{C}$ circuit, since $\mathcal{C}$ is closed under AND.

For fixed (i, j, j'), $(\text{ENC}_{j'}(x) \oplus b) \cdot R_j(x, i) \in \mathcal{C}$ has $|x| = n$ inputs and size $h(n)^{o(1)} < 2^{n^\varepsilon}$. Hence we can use our assumed #SAT algorithm to calculate $\sum_{x \in \{0,1\}^n} (\text{ENC}_{j'}(x) \oplus b) \cdot R_j(x, i)$ in time 2^{n-n^ε}. Summing over all $j \in [t]$ introduces another multiplicative factor of $h(n)^{o(1)}$. This allows us to verify the desired condition for a fixed $i, (j', b) \in S_i$. To check it for all $i, (j', b) \in S_i$ (recall $|S_i| = O(n)$ by Theorem 9) only introduces another multiplicative factor of $\text{poly}(h(n)) \cdot O(n) = \text{poly}(h(n))$ in time. Therefore the total running time for verifying consistency w.r.t. π_x is $2^{n-n^\varepsilon}\text{poly}(h(n))$.

At this point, we now know that R represents an independent set, and that R is consistent with π_x. We need to distinguish between:

1. YES case: For all x, $R(x, \cdot)$ represents an independent set of size κ.
2. NO case: For at most $2^n/g(n)$ values of $x, R(x, \cdot)$ represents an independent set of size $\geq \kappa/h(n)$.

▶ **Lemma 22.** *For all x such that $R(x, \cdot)$ represents an independent set of size a. we have $a \leq \sum_{i \in [n_2]} R(x, i) \leq at$.*

Proof. For every vertex i in the independent set, $1 \leq R(x, i) \leq t$. For all vertices i not in the independent set, we have $R(x, i) = 0$. Hence $a \leq \sum_{i \in [n_2]} R(x, i) \leq at$. ◀

Distinguishing between the YES and NO cases: To distinguish between the YES and NO cases, we now compute

$$\sum_{x \in \{0,1\}^n} \sum_{i \in [n_2]} R(x, i) \tag{3}$$

This allows us to distinguish between the YES case and NO case as:

1. YES case: We have for at least $2^n(1-1/g(n))$ values of x we have an independent set of size at most $\kappa/h(n)$. By Lemma 22 for such x, $\sum_{i\in[n_2]} R(x,i) \leq t\kappa/h(n)$. for the rest of $2^n/g(n)$ values of x the independent set could be all the vertices in the graph G_D. Hence by Lemma 22 for such values of x, $\sum_{i\in[n_2]} R(x,i) \leq tn_2 = \text{poly}(h(n))$. Hence

$$\begin{aligned}\sum_{x\in\{0,1\}^n}\sum_{i\in[n_2]} R(x,i) &\leq (2^n/g(n))\text{poly}(h(n)) + 2^n t\kappa/h(n)\\ &\leq o(2^n) + 2^n t\kappa/h(n) \quad [\text{As } h(n) = g(n)^{o(1)}]\\ &\leq o(2^n) + o(2^n\kappa) \quad [\text{As } t = h(n)^{o(1)}]\\ &\leq 2^n\kappa \quad [\text{As } \kappa > 1]\end{aligned}$$

2. NO case: We have for all $x \in \{0,1\}^n$ the independent set is at least of size κ. Hence by Lemma 22 the sum is $\sum_{x\in\{0,1\}^n}\sum_{i\in[n_2]} R(x,i) > 2^n\kappa$.

   All that remains is how to compute (3). As $R(x,i) = \sum_{j\in[t]} R_j(x,i)$, we can compute

$$\sum_{x\in\{0,1\}^n}\sum_{i\in[n_2]}\sum_{j\in[t]} R_j(x,i) = \sum_{j\in[t]}\sum_{i\in[n_2]}\sum_{x\in\{0,1\}^n} R_j(x,i)$$

For a fixed i,j, $R_j(x,i) \in \mathcal{C}$, it has $|x| = n$ inputs and size $\leq \text{poly}(h(n)^{o(1)}) = h(n)^{o(1)} < 2^{n^\varepsilon}$. Hence we can use the assumed #SAT algorithm to calculate $\sum_{x\in\{0,1\}^n} R_j(x,i)$ in time 2^{n-n^ε}. Summing over all $j \in [t], i \in [n_2]$ only introduces another $h(n)^{o(1)}\text{poly}(h(n)) = \text{poly}(h(n))$ multiplicative factor. Thus the running time for distinguishing the two cases is $2^{n-n^\varepsilon}\text{poly}(h(n))$.

In total our running time comes to $2^{n-n^\varepsilon}\text{poly}(h(n)) = 2^{n-n^{4\delta}+O(n^\delta)} \leq 2^{n-n^{3\delta}} = 2^n/g(n)^{\omega(1)}$ as $g(n) = 2^{n^{2\delta}}$ and $\varepsilon = 4\delta$. By Theorem 12, this gives us a contradiction which completes our proof. ◀

The above theorem when combined with known #SAT algorithms for $\mathsf{ACC}^0 \circ \text{THR}$ gives an quasi-NP lower bound for $\text{EMAJ} \circ \mathsf{ACC}^0 \circ \text{THR}$.

4.1 EMAJ ∘ ACC⁰ ∘ THR Lower bound

We will apply a known #SAT algorithm for $\mathsf{ACC} \circ \text{THR}$ circuits.

▶ **Theorem 23** ([22]). *For every pair of constants d, m, there exists a constant $\varepsilon \in (0,1)$ such that #SAT can be solved in time 2^{n-n^ε} time for $\mathsf{AC}^0[m] \circ$ THR circuits of depth d and size 2^{n^ε}.*

▶ **Theorem 24.** *For constants k, d, m,* quasi-NP *does not have size$(n^{\log^k n})$ EMAJ$\circ \mathsf{ACC}^0 \circ$ THR circuits of depth d.*

Proof. We first note that $\mathsf{ACC}^0 \circ \text{THR}$ is indeed typical and can represent $\text{ENC}(x)$ by $\text{poly}(n)$-sized circuits as $\text{ENC}(x) : \{0,1\}^n \to \{0,1\}^{O(n)}$ is a linear function.

By Theorem 23 we know that for all constants d there exists some constant $\epsilon \in (0,1)$ such that there exists a #SAT algorithm running in time 2^{n-n^ε} for all circuits from class $\mathsf{ACC}^0 \circ \text{THR}$ of size $\leq 2^{n^\varepsilon}$ and depth d.

The above properties imply that $\mathsf{ACC}^0 \circ \text{THR}$ satisfies the preconditions of Theorem 20 and hence for every pair of constant k, d, quasi-NP does not have size$(n^{\log^k n})$ $\text{EMAJ} \circ \mathsf{ACC}^0 \circ \text{THR}$ circuits of depth d. ◀

The above theorem can be rewritten as: For constants k, d, m, there exists a constant e such that NTIME$[n^{\log^e n}]$ does not have $n^{\log^k n}$-size EMAJ $\circ$ ACC^0 $\circ$ THR circuits of depth d. Here the constant e depends on d and m. Using a standard trick (as in [16]) this dependence can be removed as we show below.

▶ **Corollary 25.** *There exists an e such that NTIME$[n^{\log^e n}]$ does not have polynomial size EMAJ $\circ$ ACC^0 $\circ$ THR circuits.*

Proof. Assume for contradiction that for all e, there exists constants d, m such that NTIME$[n^{\log^e n}]$ has poly-sized EMAJ $\circ$ AC$^0[m]$ $\circ$ THR circuit of depth d. This implies that P has poly-sized EMAJ $\circ$ AC$^0[m]$ $\circ$ THR circuits, which further implies that *CIRCUIT EVALUATION* problem has poly-sized EMAJ $\circ$ AC$^0[m_0]$ $\circ$ THR circuit of a fixed constant depth d_0 and fixed constant m_0. Hence any circuit of size s has an equivalent poly(s)-sized EMAJ$\circ$AC$^0[m_0]\circ$THR circuit of depth d_0. Combining this with our assumption yields: For all e, there exists constants d, m such that NTIME$[n^{\log^e n}]$ has poly-sized EMAJ$\circ$AC$^0[m_0]\circ$THR circuit of depth d_0. This contradicts Theorem 24 and hence our assumption was wrong, which completes the proof. ◀

5 Extension to All Sparse Symmetric Functions

Our lower bounds extend to circuit classes of the form $f \circ \mathcal{C}$ where f denotes a family of symmetric functions that only take the value 1 on a small number of slices of the hypercube. Formally, let $f : \{0,1\}^n \to \{0,1\}$ be a symmetric function, and let $g : \{0,1,\ldots,n\} \to \{0,1\}$ be its "companion" function, where for all x, $f(x) = g(\sum_i x_i)$ (here, x_i denotes the i-th bit of x). For $k \in \{0,1,\ldots,n\}$, we say that a symmetric function f is *k-sparse* if $|g^{-1}(1)| = k$. For example, the all-zeroes function is 0-sparse, the all-ones function is n-sparse, and the EMAJ function is 1-sparse.

▶ **Theorem 26.** *Let $k < n/2$. Every k-sparse symmetric function $f : \{0,1\}^n \to \{0,1\}$ can be represented as an exact majority of $n^{O(k)}$ ANDs on k inputs.*

Proof. Given a k-sparse f and its companion function g, consider the polynomial expression

$$E(x) := \prod_{v \in g^{-1}(1)} \left(\sum_i x_i - v \right).$$

Then $E(x) = 0$ whenever $f(x) = 1$, and $E(x) \neq 0$ otherwise. Expanding E into a sum of products, we can write E as a multilinear n-variate polynomial of degree at most k, with integer coefficients of magnitude at most $n^{O(k)}$ (since each $v \leq n$). We can therefore write E as the EMAJORITY of $n^{O(k)}$ distinct ANDs on up to k inputs. ◀

The above theorem immediately implies that for every k-sparse symmetric function f_m, any circuit with an f_m at the output gate can be rewritten as a circuit with an EMAJ of fan-in at most $m^{O(k)}$ at the output gate (and ANDs of fan-in up to k below that).

▶ **Corollary 27.** *For every fixed k, and every k-sparse symmetric function family $f = \{f_n\}$,* Quasi-NP *does not have polynomial-size $f \circ$ ACC^0 $\circ$ THR circuits.*

References

1 Josh Alman, Timothy M. Chan, and R. Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In *FOCS*, pages 467–476, 2016.

2 Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. URL: `http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264`.

3 Richard Beigel, Nick Reingold, and Daniel A. Spielman. PP is closed under intersection. *J. Comput. Syst. Sci.*, 50(2):191–202, 1995. `doi:10.1006/jcss.1995.1017`.

4 Richard Beigel, Jun Tarui, and Seinosuke Toda. On probabilistic ACC circuits with an exact-threshold output gate. In *Algorithms and Computation, Third International Symposium, ISAAC '92, Nagoya, Japan, December 16-18, 1992, Proceedings*, pages 420–429, 1992.

5 Lijie Chen. Non-deterministic quasi-polynomial time is average-case hard for ACC circuits. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1281–1304, 2019.

6 Lijie Chen and R. Ryan Williams. Stronger connections between circuit analysis and circuit lower bounds, via pcps of proximity. In *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA.*, pages 19:1–19:43, 2019.

7 Ruiwen Chen, Igor Carboni Oliveira, and Rahul Santhanam. An average-case lower bound against acc^0. In *LATIN 2018: Theoretical Informatics - 13th Latin American Symposium, Buenos Aires, Argentina, April 16-19, 2018, Proceedings*, pages 317–330, 2018.

8 Irit Dinur and Omer Reingold. Assignment testers: Towards a combinatorial proof of the PCP theorem. *SIAM J. Comput.*, 36(4):975–1024, 2006. `doi:10.1137/S0097539705446962`.

9 Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost np-complete. In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 2–12, 1991.

10 Oded Goldreich. A sample of samplers: A computational perspective on sampling. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*, pages 302–332. Springer, 2011.

11 Frederic Green. A complex-number fourier technique for lower bounds on the mod-m degree. *Computational Complexity*, 9(1):16–38, 2000. `doi:10.1007/PL00001599`.

12 Kristoffer Arnsfelt Hansen. Computing symmetric boolean functions by circuits with few exact threshold gates. In *Computing and Combinatorics, 13th Annual International Conference, COCOON 2007, Banff, Canada, July 16-19, 2007, Proceedings*, pages 448–458, 2007. `doi:10.1007/978-3-540-73545-8_44`.

13 Kristoffer Arnsfelt Hansen. Depth reduction for circuits with a single layer of modular counting gates. In *Computer Science - Theory and Applications, Fourth International Computer Science Symposium in Russia, CSR 2009, Novosibirsk, Russia, August 18-23, 2009. Proceedings*, pages 117–128, 2009. `doi:10.1007/978-3-642-03351-3_13`.

14 Kristoffer Arnsfelt Hansen and Vladimir V. Podolskii. Exact threshold circuits. In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, USA, June 9-12, 2010*, pages 270–279, 2010. `doi:10.1109/CCC.2010.33`.

15 Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004. `doi:10.1007/s00037-004-0182-6`.

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

17 Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Trans. Information Theory*, 42(6):1723–1731, 1996. doi:10.1109/18.556668.

18 Suguru Tamaki. A satisfiability algorithm for depth two circuits with a sub-quadratic number of symmetric and threshold gates. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:100, 2016.

19 Nikhil Vyas and Ryan Williams. Lower bounds against sparse symmetric functions of acc circuits: Expanding the reach of #SAT algorithms, 2020. URL: https://drive.google.com/open?id=1NjlPk9FZPY3DHvxpbB7nWE3fz71E16mf.

20 Klaus W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Inf.*, 23(3):325–356, 1986. doi:10.1007/BF00289117.

21 R. Ryan Williams. New algorithms and lower bounds for circuits with linear threshold gates. *Theory of Computing*, 14(1):1–25, 2018.

22 R. Ryan Williams. New algorithms and lower bounds for circuits with linear threshold gates. *Theory of Computing*, 14(1):1–25, 2018. doi:10.4086/toc.2018.v014a017.

23 Richard Ryan Williams. Limits on representing boolean functions by linear combinations of simple functions: Thresholds, relus, and low-degree polynomials. In *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, pages 6:1–6:24, 2018. doi:10.4230/LIPIcs.CCC.2018.6.

24 Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM Journal on Computing*, 42(3):1218–1244, 2013.

25 Ryan Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014. doi:10.1145/2559903.