*Self-Improvement for Circuit-Analysis Problems*



**Citation:** Williams, R. Ryan. 2024. "Self-Improvement for Circuit-Analysis Problems."

**As Published:** 10.1145/3618260.3649723

**Publisher:** ACM

**Persistent URL:** https://hdl.handle.net/1721.1/155669

**Version:** Final published version: final published article, as it appeared in a journal, conference proceedings, or other formally published context

# Self-Improvement for Circuit-Analysis Problems*

R. Ryan Williams†
Massachusetts Institute of Technology
Cambridge, USA
rrw@mit.edu

## ABSTRACT

Many results in fine-grained complexity reveal intriguing consequences from solving various SAT problems even slightly faster than exhaustive search. We prove a *self-improving* (or "bootstrapping") theorem for Circuit-SAT, #Circuit-SAT, and its fully-quantified version: solving one of these problems faster for "large" circuit sizes implies a significant speed-up for "smaller" circuit sizes. Our general arguments work for a variety of models solving circuit-analysis problems, including non-uniform circuits and randomized models of computation.

We derive striking consequences for the complexities of these problems, in both the fine-grained and parameterized setting. For example, we show that certain fine-grained improvements on the runtime exponents of polynomial-time versions of Circuit-SAT would imply *subexponential-time* algorithms for Circuit-SAT on $2^{o(n)}$-size circuits, refuting the Exponential Time Hypothesis. We also show that any algorithm for Circuit-SAT with $k$ inputs and $n$ gates running in $1000000^k + n^{1+\varepsilon}$ time (for all $\varepsilon > 0$) would imply algorithms running in time $(1+\varepsilon)^k + n^{1+\varepsilon}$ time (for all $\varepsilon > 0$), also refuting the Exponential Time Hypothesis. Applying our ideas in the #Circuit-SAT setting, we prove new unconditional lower bounds against uniform circuits with symmetric gates for functions in deterministic linear time.

## CCS CONCEPTS

• **Theory of computation → Parameterized complexity and exact algorithms**; **Circuit complexity**; **Complexity classes**.

## KEYWORDS

bootstrapping, circuit lower bounds, circuit satisfiability, counting complexity, fine-grained complexity, quantified satisfiability

**ACM Reference Format:**
R. Ryan Williams. 2024. Self-Improvement for Circuit-Analysis Problems. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC '24), June 24–28, 2024, Vancouver, BC, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3618260.3649723

*The full version of this paper is available at https://eccc.weizmann.ac.il/report/2023/082/.

†Work was supported in part by the Simons Institute at UC Berkeley, NSF CCF-2127597, and a Frank Quick Faculty Research Innovation Fellowship.



*STOC '24, June 24–28, 2024, Vancouver, BC, Canada*



ACM ISBN 979-8-4007-0383-6/24/06
https://doi.org/10.1145/3618260.3649723

## 1 INTRODUCTION

Fine-grained complexity relates a wide array of computational problems through intricate reductions that allow us to infer tight time complexity lower bounds, based on a few hardness hypotheses. Broadly speaking, two kinds of fine-grained hypotheses have been studied, which we classify as follows.

**Weak exponent lower bounds:** These bounds assert that the optimal algorithm for a problem with a known runtime of $T(n)$ requires time at least $\Omega(T(n)^{\varepsilon})$, for some $\varepsilon > 0$. A canonical weak exponent lower bound is the Exponential Time Hypothesis:

> **ETH:** *There is an $\alpha > 0$ such that 3-SAT on $n$ variables needs $2^{\alpha n}$ time.*

Such hypotheses are often employed to argue for a conditional time lower bound in which the precise exponent is not considered as important as the *form* of the exponent; this is particularly significant for FPT algorithmics. To give two striking examples, [26] prove that the Edge Clique Cover problem, which has a simple $2^{2^k} \cdot \text{poly}(n)$ time algorithm [34], cannot be in $2^{2^{o(k)}} \cdot \text{poly}(n)$ time unless ETH is false. While it is known that approximate Nash Equilibria can be found in $n^{O(\log n)}$ time [45], it is also known [14] that an $n^{o(\log n)}$-time approximation algorithm (with "good social welfare") would contradict ETH (see also [55]).

**Strong exponent lower bounds:** These bounds assert that the optimal algorithm for a problem with a runtime of $T(n)$ requires time at least $\Omega(T(n)^{1-o(1)})$. A canonical example of a strong exponent lower bound is the Strong Exponential Time Hypothesis:

> **SETH:** *For all $\varepsilon \in (0,1)$, there is a $k$ such that $k$-SAT on $n$ variables needs $2^{n(1-\varepsilon)}$ time.*

Such hypotheses are generally used to argue that the best-known running time for a problem is optimal up to low-order terms (see [64] for a large sample of reductions and problems).

It is intuitively obvious that a strong exponent lower bound is indeed a stronger assumption than a weak exponent lower bound: for example, SETH implies ETH [18, 36]. Conversely, the question of whether ETH implies SETH is a major open problem (already raised explicitly in [35]). It is entirely uncertain how such an implication might be proved. In this paper, we ask a more general question:

> **Question:** *Can weak exponent lower bounds be "amplified" into strong exponent lower bounds?*

A positive answer to the question amounts to a situation where improving slightly on the running time exponent of one problem leads to an *arbitrary* polynomial improvement in the best-known time exponent of another problem. We will prove a result of this form for a "large" variant of the Circuit SAT problem, as well as its counting and quantified variants.[1]

[1]See the end of the Introduction for an alternative viewpoint.

We begin with the following version of Circuit SAT, where the input circuit is set to be so large that the problem is polynomial-time solvable. Let $\varepsilon \in (0,1]$ be a (small) constant parameter.

**Problem:** Large Circuit SAT
**Given:** A circuit $C$ with at most $n = \varepsilon \log(N)$ inputs and $N$ gates (a.k.a. $N$ size).[2]
**Decide:** Is there an $a \in \{0,1\}^n$ such that $C(a) = 1$?

For $\varepsilon \leq 1$, the circuit instances of Large Circuit SAT are so large that they cannot possibly be minimal: recall that the maximum circuit complexity of any $\varepsilon \log(N)$-input function is $o(N^{\varepsilon})$ [37]. Such a large circuit must therefore have enormously redundant parts that could potentially be simplified, in a satisfiability algorithm. Intuitively, Circuit SAT can only get *easier* to solve as the circuit size increases (this corresponds to decreasing $\varepsilon$).

Observe the brute-force algorithm for Large Circuit SAT takes $\tilde{O}(N^{1+\varepsilon})$ steps. Can we improve upon the brute-force algorithm for Large Circuit SAT? Can the obvious $N^{1+\varepsilon}$ time algorithm for Large Circuit SAT be reduced to $N^{1+o(1)}$ for *some constant* $\varepsilon > 0$?

A corollary of our main result is that such an algorithm would already imply that the Exponential Time Hypothesis is false: in fact, the existence of such an algorithm implies that Circuit SAT on $2^{o(n)}$-size circuits can be solved in $2^{\varepsilon n}$ time for every $\varepsilon > 0$. The most general form of our connection is the following.

Theorem 1.1 (Circuit-SAT "Self-Improvement", Section 3). *Let $\alpha, \beta > 0$, with $\alpha \leq \beta$. Suppose* Circuit SAT *on $2^{\alpha n + o(n)}$-size circuits can be solved in $2^{\beta n + o(n)}$ time. Then* Circuit SAT *on $2^{o(n)}$-size circuits can be solved in $2^{(\beta-\alpha)n+o(n)}$ time.*

We call such a result *self-improving*, as it proceeds by an induction where in each stage of induction, the running time of the SAT algorithm for $2^{o(n)}$-size circuits is improved by combining the assumed algorithm for Large Circuit SAT with the SAT algorithm derived in the previous stage. Theorem 1.1 holds for any computational model such that $T$-time algorithms can be simulated by circuits of size $T^{1+o(1)}$ (for example, multitape Turing machines [54]).[3] Theorem 1.1 also holds for randomized algorithms (see the full version of the paper [68]) as well as for non-uniform models of computation: given $2^{\beta n+o(n)}$-size circuits solving Circuit SAT on $2^{\alpha n+o(n)}$-size inputs, we can construct $2^{\beta-\alpha+o(n)}$-size circuits for Circuit SAT on $2^{o(n)}$-size inputs. The following is an immediate corollary of Theorem 1.1.

Corollary 1.2 (ETH Versus Large Circuit SAT). *ETH implies that, for* **every** $\varepsilon > 0$, Large Circuit SAT *with $\varepsilon \log(N)$ inputs is not solvable in $N^{1+o(1)}$ time.*

In fact, we only have to assume there is an $\varepsilon > 0$ such that Circuit-SAT on $2^{o(n)}$-size circuits cannot be solved in $2^{\varepsilon n+o(n)}$ time, which is (presumably) a significantly weaker hypothesis than ETH itself, which is only concerned with the complexity of $k$-SAT. The upshot is that, from a weak-exponent lower bound hypothesis like ETH, we obtain a lower bound of a strong-exponential character for a polynomial-time solvable problem: if the brute-force $\tilde{O}(N^{1+\varepsilon})$-time algorithm for Large Circuit SAT can be improved to $N^{1+o(1)}$ time, for *any* $\varepsilon > 0$, then we obtain an arbitrary polynomial improvement over exhaustive search for Circuit SAT on "small" circuits.

It is also instructive to compare Corollary 1.2 with the implications obtained by assuming a Circuit SAT form of SETH, rather than ETH:

Corollary 1.3 (SETH Versus Large Circuit SAT). *Assume that for every $\varepsilon > 0$,* Circuit SAT *on $2^{o(n)}$-size circuits cannot be solved in $2^{(1-\varepsilon)n}$ time. Then for every $\alpha \geq 0$ and every $\varepsilon > 0$,* Circuit SAT *on $2^{\alpha n+o(n)}$-size circuits cannot be solved in time $2^{\alpha n+(1-\varepsilon)n+o(n)}$.*

That is, if brute-force search is essentially optimal for solving Circuit SAT on subexponential-size circuits, then brute-force is also optimal for solving Circuit SAT on arbitrarily large $2^{O(n)}$-size circuits, in spite of the fact that the "large" circuit-size case can easily be reduced to the "small" case by adding extra (non-functional) inputs (see Theorem 1.4). Phrasing Corollary 1.3 another way, we can say that if there is an $\varepsilon > 0$ and a $\delta \in (0,1)$ such that Circuit SAT on $N$-size circuits with $\varepsilon \log(N)$ inputs can be solved in $N^{1+\delta\varepsilon}$ time (for example), then Circuit SAT on $2^{o(n)}$-size circuits can be solved in $2^{\delta n+o(n)}$ time.

*An Equivalence.* The ideas of theorem 1.1 lead to a surprising equivalence for solving Large Circuit SAT efficiently. For simplicity, we state the result in terms of algorithms solving Circuit SAT, but it also applies to non-uniform and randomized algorithms (see the full version for details [68]). Let $\varepsilon$-Large Circuit SAT be the problem of checking satisfiability for circuits of size $N$ with $\varepsilon \log(N)$ inputs.

Theorem 1.4 (Section 3). *The following are equivalent:*

*(1) There is* **some** $\varepsilon \in (0,1)$ *such that* $\varepsilon$-Large Circuit SAT *is in $N^{1+o(1)}$ time.*
*(2) For* **every** $\alpha > 0$ *(including arbitrarily large $\alpha$),* $\alpha$-Large Circuit SAT *is in $N^{1+o(1)}$ time.*

Theorem 1.4 shows an existentially-quantified statement is **equivalent** to its corresponding universally-quantified statement: if we can solve Circuit SAT on $n$-input $2^{Kn}$-size in $2^{Kn+o(n)}$ time, for *some* constant $K > 0$, then an analogous algorithm exists for **every** $K > 0$. As a consequence, the hypothesis would imply that Circuit SAT on $2^{\varepsilon n}$-size circuits (for any tiny $\varepsilon > 0$) can also be solved in $2^{\varepsilon n+o(n)}$, refuting the (circuit version of) ETH. Therefore, Theorem 1.4 can be seen as a strengthening of Corollary 1.2. In fact, an even stronger equivalence holds, between nearly-linear-time algorithms for Circuit SAT on $\varepsilon \log(N)$ inputs for arbitrarily small $\varepsilon > 0$, and extensions of Circuit SAT that correspond to levels of the polynomial hierarchy (see Theorem 3.4).

*Self-Improvement for #SAT and QBF..* The proofs of Theorem 1.1 and Theorem 1.4 are quite general. We show that analogous self-improvement results hold for #Circuit SAT, where we wish to *count* the number of SAT assignments to a given circuit, as well as Q-Circuit SAT, the quantified version of Circuit SAT, where we

[2] The circuits can be over any universal basis of constant fan-in, e.g., AND/OR/NOT.
[3] We discuss in the full version [68] how to obtain results for more powerful models of computation, like random access machines. Intuitively, we just have to change Circuit SAT to a satisfiability problem with a suitable predicate, e.g., Ram Sat for random access machines.

are given a fully-quantified sentence of the form

$$(Q_1\, x_1) \cdots (Q_n\, x_n)[C(x_1, \ldots, x_n)],$$

where each $Q_i \in \{\exists, \forall\}$, $C$ is a circuit, and we wish to decide if the sentence is true or false.

Theorem 1.5. *Theorem 1.1 holds for the* #Circuit SAT *problem and* Q-Circuit SAT*, in place of* Circuit SAT.

In fact, all consequences stated for Circuit SAT carry over for #Circuit SAT and Q-Circuit SAT.

*An FFT for Circuits Would Refute Exponential-Time Hypotheses.* A major application of the Fast Fourier Transform (FFT) [24] is that univariate degree-$n$ polynomials over a field can be evaluated on any $n$ points in $n \cdot \text{poly}(\log n)$ operations [13, 30], a great improvement over the obvious $\Theta(n^2)$ algorithm. Recent work has extended this fundamental result to the multivariate setting [10, 11, 33, 40].

Should we expect fast multipoint evaluation for more complex computational models, such as Boolean circuits? On the one hand, an old result of W. J. Paul ([53], Lemma 2) gives an efficient circuit $C$ for multipoint evaluation of Boolean functions: given $x_1, \ldots, x_k \in \{0,1\}^n$ and the truth table $T \in \{0,1\}^{2^n}$ of a function $f : \{0,1\}^n \to \{0,1\}$, we have $C(x_1, \ldots, x_k, T) = (f(x_1), \ldots, f(x_k))$, for a circuit $C$ of size only $\text{poly}(n) \cdot (2^n + k)$. Thus, for very hard functions (that cannot be represented much smaller than their $2^n$ truth table), there are circuits for multipoint evaluation with size about $k + 2^n$, improving over the obvious $k2^n$ bound. On the other hand, standard results in fine-grained complexity show that if the truth tables of size-$s$ (unrestricted) circuits could be computed in time $\text{poly}(n) \cdot (s + 2^n)$ (for example), then SETH and the 3SUM conjecture are false.[4] An immediate corollary of Theorem 1.1 and Theorem 1.5 is that significantly weaker hypotheses suffice:

Corollary 1.6. *If $n$-input circuits of size $s$ can be evaluated on all inputs in $2^{n+o(n)} + s^{1+o(1)}$ time, then the circuit versions of #ETH [27] and the quantified version of ETH [23] are false:* #Circuit SAT *and* Q-Circuit SAT *on $n$-input $2^{o(n)}$-size circuits are both in $2^{\varepsilon n}$ time, for all $\varepsilon > 0$.*

That is, the difficulty of finding an FFT-like algorithm for fast multipoint circuit evaluation can be based on much weaker hypotheses than SETH, weaker than even circuit versions of SETH (used to argue for the hardness of problems like Edit Distance [1]).

*A Parameterized Complexity Counterpart.* Another version of Theorem 1.4 can be stated in the framework of parameterized complexity, yielding another type of surprising equivalence. Letting $k$ be the number of variables as a parameter, and letting $n$ be the circuit size, brute force yields a $2^k \cdot n\, \text{poly}\log(n)$ time algorithm for Circuit SAT. A standard trick in parameterized complexity [25, 29] implies that for every $\varepsilon > 0$, there is some constant $c > 1$ such that Circuit SAT can be solved in $O(c^k + n^{1+\varepsilon})$ time.[5]

Could one reverse the order of the quantifiers in this statement? Could there be a **universal** $c > 1$ such that for all $\varepsilon > 0$, Circuit SAT can be solved in $O(c^k + n^{1+\varepsilon})$ time? (Another way of phrasing the question: could we replace $2^k \cdot n^{1+\varepsilon}$ time, with $c^k + n^{1+\varepsilon}$ time for a large $c$?) We show that such an algorithm would in fact disprove the Exponential Time Hypothesis:

Theorem 1.7 (Section 4). *There is a $c > 1$ such that for all $\varepsilon > 0$,* Circuit SAT *is in $O(c^k + n^{1+\varepsilon})$ time if and only if* **for every** *$c > 1$ and $\varepsilon > 0$,* Circuit SAT *is in $O(c^k + n^{1+\varepsilon})$ time.*

For example, from an algorithm running in $O(10000000^k + n^{1+\varepsilon})$ time for Circuit SAT, for all $\varepsilon > 0$, we could derive a Circuit SAT algorithm running in $(1+\varepsilon)^k + n^{1+\varepsilon}$ time, for all $\varepsilon > 0$. Philosophically, Theorem 1.7 may be viewed as more of a "true self-improvement" than other results, as we really are improving the running time of Circuit SAT to an arbitrarily small exponential bound, starting from a certain type of exponential-time algorithm for the same Circuit SAT problem.

*A Uniform Circuit Lower Bound for Linear Time.* Studying the consequences of faster #Circuit SAT algorithms for large circuits further, we prove new unconditional lower bounds against uniform circuit classes, where fast multipoint evaluation algorithms exist (and thereby small improvements over exhaustive search are also possible). Let $\mathsf{SYM} \circ \mathsf{SYM}$ denote the class of Boolean circuits which are depth-two circuits comprised of arbitrary Boolean symmetric functions (with unbounded fan-in). $\mathsf{SYM} \circ \mathsf{SYM}$ is one of those natural "weak-looking" circuit classes for which the known lower bounds are surprisingly meager. In terms of non-uniform lower bounds against $\mathsf{SYM} \circ \mathsf{SYM}$, it is only known that there are functions in $\mathsf{E}^{\mathsf{NP}}$ which do not have non-uniform $\mathsf{SYM} \circ \mathsf{SYM}$ circuits of $n^{2-\varepsilon}$ gates, for all $\varepsilon > 0$ [8, 59]. Since $\mathsf{SYM} \circ \mathsf{SYM}$ can be simulated in depth-3 $\mathsf{TC}^0$ with a polynomial blowup in size, one can deduce from known results on $\mathsf{TC}^0$ ([5]) that the Permanent does not have polynomial-size *highly-uniform* $\mathsf{SYM} \circ \mathsf{SYM}$ circuits. It also follows from the literature that, for some $\alpha > 0$, SAT does not have highly-uniform $\mathsf{SYM} \circ \mathsf{SYM}$ circuits with $n^{1+\alpha}$ gates [6].[6]

In the full version of the paper, we prove a super-linear gate lower bound for computing problems in linear time with uniform $\mathsf{SYM} \circ \mathsf{SYM}$ circuits.

Theorem 1.8 ([68]). *There are linear-time decision problems which do not have POLYLOGTIME-uniform $\mathsf{SYM} \circ \mathsf{SYM}$ circuits of $n^c$ gates, for all $c < 1.199$.*

(For an explicit problem exhibiting the lower bound, one could take the P-complete Circuit Eval decision problem.) The proof of Theorem 1.8 has the form of an indirect diagonalization: assuming the opposite, we derive a simulation of time-bounded computation contradicting hierarchy theorem. However, for all prior such lower bounds that we are aware of, across a variety of models (such as [5, 7, 32, 39, 48, 61–63]), the proofs require that the hard function is *much* harder than linear-time computable. For example, the time-space tradeoffs for *SAT* [16, 32, 61] crucially require that the hard function is NP-hard under highly local reductions.

[4] For example, see footnote 7 in [66].

[5] Let $\varepsilon > 0$ be given and let $n$ be sufficiently large. If $2^k < n^{\varepsilon/2}$ then $2^k \cdot n^{1+o(1)} < n^{1+\varepsilon}$. Otherwise, $2^k \geq n^{\varepsilon/2}$, i.e., $\log(n) \leq 2k/\varepsilon$. Setting $c = 2^{1+(2+\varepsilon)/\varepsilon}$, we have $2^k \cdot n^{1+o(1)} < 2^k \cdot n^{1+\varepsilon/2} = 2^k \cdot 2^{(1+\varepsilon/2)\log n} \leq 2^{k+(1+\varepsilon/2)\cdot 2k/\varepsilon} = c^k$.

[6] Note that, although it is also known [38] that there are functions in P that require $\Omega(n^{1.5-o(1)})$ gates to be computed by non-uniform depth-3 $\mathsf{TC}^0$ circuits, the translation of $\mathsf{SYM} \circ \mathsf{SYM}$ into depth-3 $\mathsf{TC}^0$ can increase the total number of gates by a factor of $n$, so the methods of [38] do not directly yield linear gate lower bounds for $\mathsf{SYM} \circ \mathsf{SYM}$ circuits. Certainly the random restriction lemmas of [38] do not directly apply either, since PARITY is a type of symmetric gate, and is immune to restrictions.

We deduce a contradiction by exploiting *circuit-analysis algorithms* for SYM∘SYM. That is, we establish a version of the "algorithmic method" for circuit lower bounds (initiated by Williams [66, 67]) that applies to *uniform* circuits, and allows the hard function to be contained in P.[7] We apply the assumption-to-be-contradicted in two different ways: once on an initial $2^n$-time computation (that we wish to speed up), and again on a circuit that counts the number of SYM gates that are true on the bottom layer, using the POLYLOGTIME-uniform algorithm for generating the gates on the bottom layer. In the end, our contradictory simulation is achieved by applying fast rectangular matrix multiplication [44] appropriately to "speed-up" the evaluation of a SYM ∘ SYM circuit. If the matrix multiplication exponent $\omega$ happens to be 2, our gate lower bound would improve to $n^{1.36}$.

Indeed, matrix multiplication allows us to compute truth tables of SYM ∘ SYM circuits faster than the obvious algorithm, and our proof demonstrates how such an algorithm can be used to establish new lower bounds for linear-time computation. This addresses a question of Williams [69], who gave a faster truth-table evaluation algorithm for THR ∘ THR circuits, and asked if such algorithms suffice for deriving lower bounds. (However, super-linear gate lower bounds against THR ∘ THR are already known; see [38]. Thus we instead state our results in terms of SYM ∘ SYM.) One can think of our approach as trading non-uniformity in the circuit lower bound for a significant reduction in the complexity of the hard function (from $\mathsf{E}^{\mathsf{NP}}$ or QuasiNP, down to linear time).

*On the Difficulty of Further Improving Self-Improvement.* We have shown that self-improvement of Circuit SAT is possible for deterministic, randomized, and non-uniform algorithms; what about other computational models, such as nondeterministic machines and those in the polynomial hierarchy? We show that such questions have an intimate connection to the NP versus $\mathsf{NC}^1$ problem. We already mentioned (Theorem 1.5) that self-improvement holds for the Q-Circuit SAT (Quantified Circuit SAT) problem, for deterministic and randomized algorithms. Observe that, if we allow our algorithms to call an oracle in the polynomial hierarchy, then Q-Circuit SAT *can* be decided efficiently.

Proposition 1. *For all positive real $\alpha > 0$, Q-Circuit SAT on $2^{\alpha n}$-size circuits can be decided in poly$(n) \cdot (2^{\alpha n} + 2^n)$ time with a $\Sigma_2 SAT$ oracle.*

Indeed, with a $\Sigma_2$ machine, one can simply guess the $2^n$-bit truth table of the given circuit, universally verify the truth table is correct on all inputs, and verify that the QBF defined on the truth table is true, in $O(2^{\alpha n} + 2^n)$ time. (We can use $\Sigma_2 SAT$ as an oracle specifically, because of tight reductions from $\Sigma_2$ time $T(n)$ to $\Sigma_2 SAT$; see for example [32].) This observation naturally begs the question of whether Q-Circuit SAT self-improvement is possible on algorithms with an oracle in $\Sigma_2$P. In the full version, we show that such a result would separate NP from $\mathsf{NC}^1$, even if we could only obtain non-uniform circuits as a consequence.

Theorem 1.9 ([68]). *Suppose a self-improvement result holds for Q-Circuit SAT with $\Sigma_2$P-oracle algorithms, i.e., assume:*

*There is some $k > 0$ such that Q-Circuit SAT on $2^{kn+o(n)}$-size circuits in $2^{kn+o(n)}$ time (with a $\Sigma_2 SAT$ oracle) implies that for all $\varepsilon > 0$, Q-Circuit SAT on $2^{o(n)}$-size circuits has $2^{\varepsilon n+o(n)}$-size non-uniform $\Sigma_2 SAT$-oracle circuits.*

*Then* NP $\neq$ $\mathsf{NC}^1$.

(Here, we use the LOGTIME-uniform definition of $\mathsf{NC}^1$ [65].) The choice of "$\Sigma_2$" in the theorem statement is somewhat arbitrary: using "$\Sigma_c$" for any $c \geq 2$ would suffice. To prove this theorem, we show that NP = $\mathsf{NC}^1$ implies a strong circuit lower bound on Q-Circuit SAT, even for circuits with an oracle in the polynomial hierarchy.

In the full version [68], we also prove as a consequence of self-improvement that, if there is any $k > 0$ such that Q-Circuit SAT on $2^{kn}$-size circuit predicates has non-uniform circuits of size $2^{kn+o(n)}$, then NP $\neq$ $\mathsf{NC}^1$. We should stress that we do not (yet) consider these theorems as a viable approach to separating NP from $\mathsf{NC}^1$, but they do elevate the question of *why* self-improvement only seems to work for deterministic and randomized algorithms, but not for stronger models of computation.

*On the Meaning of This Work.* Does this paper prove a connection between "weak exponential" lower bounds for one problem $\Pi$ and "strong exponential" lower bounds for another problem $\Psi$? It is the opinion of the author that the answer is yes, and that Theorem 1.4 is the most striking example of the connection: an $N^{1+o(1)}$-time algorithm for Circuit SAT on $\varepsilon \log(N)$ inputs (for **any** $\varepsilon > 0$) is equivalent to having an $N^{1+o(1)}$-time algorithm for Circuit SAT on $K \log(N)$ inputs, for **every** $K$, no matter how large. However, others may disagree, and argue that what is actually being proved here is an equivalence between *different exponential-time hypotheses*: one is showing that the "exponential-time hypothesis for Circuit SAT on $2^{Kn}$-size circuits" for arbitrarily large $K \geq 1$ (defined in the appropriate way) is equivalent to the "exponential-time hypothesis for Circuit SAT on $2^{\varepsilon n}$-size circuits" for arbitrarily small $\varepsilon > 0$. The author believes this is also a perfectly valid interpretation of the results; the difference boils down to what one counts as "weak exponential" versus what is "strong exponential". (Both interpretations are interesting, in the opinion of the author.) The parameterized version of our self-improvement result (Theorem 1.7) also shows what may be considered a "truer" self-improvement: one time bound for Circuit SAT directly implies a strictly stronger time bound for Circuit SAT.

## 2 PRELIMINARIES

We assume familiarity with computational complexity, especially circuit complexity [9, 37, 65]. We are often interested in LOGTIME-uniform (and POLYLOGTIME-uniform circuits, respectively), where local information about the gates of poly$(n)$-size circuits can be determined in time linear (respectively, polynomial) in the names of the gates, each of which take $O(\log n)$ bits to describe. We will give technical details on such uniformity conditions as needed in our proofs; see [65] for full technical definitions.

*Notation and Defaults.* Unless otherwise specified, our Boolean circuits are over the basis of all possible gates of fan-in two (the

[7] See [57] for another proposal, which would apply to uniform lower bounds for NP and PSPACE if it can be realized. Santhanam's approach looks significantly more general than ours, but does not seem to extend down to functions in P.

particular gate basis will not matter for our results, as long as the basis is universal and each gate has constant fan-in.)

As is standard for bounded fan-in circuits, the *size* of a circuit is defined to be the number of gates. For a given circuit $C$, we let $\langle C \rangle$ denote the description of $C$ in binary.

Recall that Circuit Eval is the P-complete problem of Circuit Evaluation, in which we are given the description $\langle C \rangle$ of a circuit $C$, and an assignment $a$ to the inputs of $C$, and wish to output $C(a) = 1$. For notational convenience, in this paper we redefine Circuit Eval to be the following multi-output problem:

> Circuit Eval: *Given the description $\langle C \rangle$ of a circuit $C$, and a* **partial** *assignment $a$ to the inputs of $C$, output the description of the circuit $C'(x) := C(a, x)$, where $x$ denotes the remaining unassigned inputs of $C$.*

The following basic fact about circuit evaluation will be very useful.

Lemma 2.1 (Valiant [60], Pippenger-Fischer [54]). *The problem Circuit Eval has circuits of size $\tilde{O}(n)$, constructible in $\tilde{O}(n)$ time (even on a multitape Turing machine). In particular, for every $n$, there is a circuit $D_n$ of $\tilde{O}(n)$ size such that, given the description $\langle C \rangle$ of a circuit $C$ of size $n$ and a partial assignment $a$ to some of $C$'s inputs, $D_n(\langle C \rangle, a)$ outputs a description of $C$ restricted to the partial assignment $a$.*

## 2.1 Related Work

The most directly relevant prior result is that of Salamon and Wehar [56], who show if Circuit SAT with $2^n$ gates and $n$ inputs is solvable in $2^{n+o(n)}$ time, then Circuit SAT with $m$ gates is solvable in $2^{\varepsilon m}$ time for every $\varepsilon > 0$. Our results can be seen as substantial generalizations, weakening the hypotheses and strengthening the resulting conclusions. More precisely, they require that an $\tilde{O}(N^2)$-time solvable version of Circuit SAT can be improved to $N^{1+o(1)}$ time, in order to get a subexponential-time algorithm for satisfiability of $O(n)$-size circuits. In contrast, one corollary of our main result (Corollary 1.2) states that improving an $\tilde{O}(N^{1+\varepsilon})$-time solvable version of Circuit SAT to $N^{1+o(1)}$ time, for **any** $\varepsilon > 0$, implies a subexponential-time algorithm for satisfiability of *subexponential*-size circuits. (Indeed, following Theorem 3.4, we obtain a subexponential-time algorithm for $\Sigma_k$Circuit SAT, for every constant $k$.) At a high level, the approach of Salamon and Wehar looks similar: they partition the variables of their input circuit, and call an algorithm on restrictions of the input circuit based on the variable partition. However, their approach appears to require the use of a specialized computational model (which they call "DTIWI") that hampers the generality of what they can prove.

Other works have demonstrated phenomena which are similar to our self-improvement results, but differ in various critical ways. Williams [66] studied the consequences of speeding-up exhaustive search in limited scenarios. Along with showing that slightly faster Circuit SAT algorithms imply non-uniform circuit lower bounds, he also showed that if every problem $\Pi$ solvable with $\log n$ bits of nondeterminism in $n^c$ time and $(\log n)^d$ space can be simulated in $O(n^{c+0.99})$ time and $\text{poly}(\log n)^d$ space for all $c, d \geq 1$, then a dramatic speed-up is possible: every such $\Pi$ can be solved nondeterministically in $O(n^3)$ time, which would imply LOGSPACE $\neq$ NP among other consequences. Thus, by imposing a space restriction on the verifier and the assumed simulation, a more dramatic simulation is possible from assuming a minor speed-up. Williams also observes that a Circuit SAT algorithm running in $4^{(1-\varepsilon)n}$ time on circuits of size $2^n$ with $n$ inputs, for some $\varepsilon > 0$, implies that the 3SUM conjecture is false.[8] It is also easy to see that the same hypothesis implies that SETH and thereby the Orthogonal Vectors Conjecture is false (this also follows from the base case of Theorem 1.1).

Paturi and Pudlák [52] study OPP algorithms, which are probabilistic polynomial time algorithms with $1/p(n)$ success probability, where $p(n)$ can be exponential. They show if Circuit SAT has an OPP algorithm with success probability $1.999^{-n}$, then Circuit SAT on poly$(n)$-size circuits has deterministic circuits of size $2^{n^{1-\varepsilon}}$ for some $\varepsilon > 0$. Their argument involves applying the polynomial-size circuit for Circuit SAT to itself in an interesting way. While their Circuit SAT conclusion seems stronger than the ones we derive (we derive $2^{\varepsilon n}$-time algorithms for $2^{o(n)}$-size circuits), their Circuit SAT hypothesis looks stronger than the hypotheses that we consider, especially for our extensions to randomized algorithms for Circuit SAT (found in the full version of the paper).

The results in this paper show how "minor-looking" algorithmic improvements would imply major algorithmic improvements, in which the minor algorithm is repeatedly applied to achieve faster algorithms on smaller input lengths. These results can be seen as converses of hardness magnification phenomena [6, 20–22, 46, 47, 49, 50, 58], in which "minor-looking" computational lower bounds would imply major lower bounds. The contrapositives of hardness magnification results can also be viewed in a similar light. For example, Allender-Koucky [6] show that if Boolean Formula Evaluation has constant-depth MAJORITY/NOT ($\mathsf{TC}^0$) circuits of *any* polynomial size, then the problem also has $O(1/\varepsilon)$-depth MAJORITY/NOT circuits of $n^{1+\varepsilon}$ size, for all $\varepsilon > 0$. This is proved by exploiting the nice downward self-reducibility of Formula Evaluation. Our setting appears to be very different from that of hardness magnification. We study versions of NP-hard problems in a "polynomial-time solvable" regime, and show that sufficiently strong algorithms in this setting would imply exponentially-faster algorithms in the "super-polynomial-time solvable" regime. In our self-improvement results for Circuit SAT, #Circuit SAT, and Q-Circuit SAT, the main property required is that the problem is "embarrassingly parallel", in that the space of variable assignments can be partitioned in a simple way so that the overall answer can be easily obtained from the answers on the parts.

In general, when one considers Circuit SAT on circuits which are large relative to the number of input variables, one is studying a problem with "bounded nondeterminism" or "limited nondeterminism", where the amount of nondeterminism is significantly less than the input length $n$ (in our case, the amount of nondeterminism is $O(\log n)$). The theory of complexity classes with limited nondeterminism was initiated in [41]; further related references include [12, 15, 17, 31, 66].

[8]Recall the 3SUM problem asks: *given a set $S$ of $n$ numbers, are there three which sum to zero?* The 3SUM conjecture is that there is no $n^{2-\varepsilon}$ time algorithm for 3SUM, where $\varepsilon > 0$.

Finally, we note that a type of self-improvement was known in algebraic complexity. In particular, there are bootstrapping results are known for derandomizing the Polynomial Identity Testing (PIT) problem [3, 42, 43]. Roughly speaking, these results show that "minor" improvements over the obvious deterministic black-box PIT algorithm for circuits with a constant number of variables would yield a nearly-polynomial-time deterministic algorithm for the full PIT problem. The proofs of these results also work in stages, where in each successive stage, the algorithm from the previous stage is used to build a faster algorithm.

# 3 SELF-IMPROVEMENT FOR CIRCUIT ANALYSIS PROBLEMS

Here, we prove the main self-improvement result, showing how non-trivial algorithms for Large Circuit SAT would imply faster algorithms for Circuit SAT on subexponential-size circuits.

**Reminder of Theorem 1.1.** *Let $\alpha, \beta$ be positive reals, with $\alpha \leq \beta$. Suppose* Circuit SAT *on $2^{\alpha n+o(n)}$-size circuits can be solved in $2^{\beta n+o(n)}$ time. Then* Circuit SAT *on $2^{o(n)}$-size circuits can be solved in $2^{(\beta-\alpha)n+o(n)}$ time.*

Before we begin the proof, the following intuition may be helpful. Suppose we have a circuit $C$ of size $2^{o(n)}$ and $n$ inputs, and we want to solve Circuit-SAT on $C$, as fast as possible. Furthermore, assume we have in our hands an algorithm for Circuit-SAT that runs on circuits of size $2^{n'+o(n)}$ with $n'$ inputs, and this algorithm runs in $2^{n'+o(n)}$ time.

To get a faster SAT algorithm for $C$ using the assumed algorithm, we may start by offloading some of the work of satisfiability onto the circuit itself, with the following "OR trick" used in several fine-grained algorithms [2, 19, 67]. WLOG suppose $n$ is even. Take the first $n/2$ inputs of $C$, and consider the circuit $C'$ on $n/2$ inputs, defined as follows:

$$C'(x_1, \ldots, x_{n/2}) = \bigvee_{(a_1,\ldots,a_{n/2})\in\{0,1\}^{2^{n/2}}} C(a_1, \ldots, a_{n/2}, x_1, \ldots, x_{n/2}).$$

That is, $C'$ takes an OR over all possible assignments to the first $n' := n/2$ variables of $C$, plugging each assignment into a separate copy of $C$. Observe that $C'$ has $2^{n'+o(n')}$ size, and $C'$ is satisfiable if and only if $C$ is satisfiable. Now, if we apply our assumed SAT algorithm to $C'$, we get a new SAT algorithm for $2^{o(n)}$-size $C$ that runs in only $2^{n'+o(n')} = 2^{n/2+o(n)}$ time, beating the brute-force algorithm which runs in $2^{n+o(n)}$ time.

Our key observation is that the new SAT algorithm just derived can be combined with the assumed SAT algorithm, to "improve upon itself". After we split the variables into two parts, instead of taking an OR over all possible assignments, we can run the new $2^{n/2+o(n)}$-time SAT algorithm for $2^{o(n)}$-size circuits. For example, suppose we split the $n$ variables into an "outer" set of $n/3$ and an "inner" set of $2n/3$. After any assignment to the outer variables is made, the remaining SAT instance on $n' = 2n/3$ variables can be solved in $2^{n'/2+o(n')} = 2^{n/3+o(n)}$ time, using our new SAT algorithm. Therefore, by calling our assumed SAT algorithm *on a $2^{n/3+o(n)}$-size circuit that encodes the new SAT algorithm, with the $n/3$ outer variables as input*, we can derive an even faster SAT algorithm, running in $2^{n/3+o(n)}$ time on $2^{o(n)}$-size circuits. Repeating the argument, we can achieve $2^{n/k+o(n)}$ time for any constant $k \geq 1$. The following proof is a very general form of this intuition.

Proof. We will inductively show that for every $\varepsilon > 0$, there is an algorithm which can decide satisfiability for $2^{o(n)}$-size circuits in $2^{\varepsilon n}$ time.

Start with a Circuit SAT instance $C$ of $2^{o(n)}$ size and $n$ inputs. Let $S$ be an algorithm running in $2^{\beta n+o(n)}$ time that takes as input the description $\langle C'\rangle$ of a $2^{\alpha n+o(n)}$-size circuit $C'$, and determines satisfiability for $C'$.

Our first improved algorithm $\mathcal{F}_1$ can be described as follows. Given $\langle C\rangle$, the algorithm constructs a circuit $D$ on $m = n/(1+\alpha)$ inputs with the following behavior. First, given an assignment $x$ of $m = n/(1+\alpha)$ bits, $D$ feeds the bits of $x$ into the first $n/(1+\alpha)$ inputs of $C$. Formally, this is implemented by calling Circuit Eval$(\langle C\rangle, x)$. (The circuit $D$ has the description of $C$ hard-coded.) By Lemma 2.1, Circuit Eval$(\langle C\rangle, x)$ outputs the description $\langle C'\rangle$ of a circuit $C'$ with $m' = \alpha n/(1+\alpha)$ inputs and $2^{o(n)} \leq 2^{o(m)}$ size. Next, $D$ enumerates all possible $2^{m'}$ assignments to the $m'$ inputs of $C'$, and takes the OR over all such assignments. Thus, $D$ has size

$$2^{m'+o(m)} \leq 2^{\alpha n/(1+\alpha)+o(n)} \leq 2^{\alpha m+o(m)},$$

and has $m$ inputs. Note that a description $\langle D\rangle$ of $D$ can be constructed in $2^{\alpha m+o(m)}$ time: we only have to write down a description of the OR of $2^{m'}$ circuits of the form Circuit Eval$(\langle C'\rangle, a)$, over all possible $a \in \{0,1\}^{m'}$. Furthermore, observe that $D$ has $m = n/(1+\alpha)$ inputs.

Finally, the algorithm feeds the description $\langle D\rangle$ of size $2^{\alpha m+o(m)}$ to the assumed algorithm $S$, which runs in time

$$2^{\beta m+o(m)} \leq 2^{\beta n/(1+\alpha)+o(n)},$$

and outputs a yes/no answer. Observe that $C$ is satisfiable if and only if $S(\langle D\rangle)$ outputs yes: there is a satisfying assignment to $C$ if and only if there is a partial assignment $a \in \{0,1\}^{m'}$ such that Circuit Eval$(\langle C'\rangle, a)$ is satisfiable, which is true if and only if $S(\langle D\rangle)$ outputs yes.

From the above, we conclude that satisfiability of circuits of $2^{o(n)}$ size can be determined in $2^{\beta n/(1+\alpha)+o(n)}$ time. Denote this algorithm by $\mathcal{F}_1$.

We can repeat the above argument, but instead of enumerating all possible assignments (simulating brute-force search), we call the algorithm $\mathcal{F}_1$ instead. Suppose inductively that satisfiability of circuits of $n$ inputs and $2^{o(n)}$ size can be determined by an algorithm $\mathcal{F}_k$ running in $2^{f_k n+o(n)}$ time. (For instance, in the base case, we know we can set $f_0 := \beta$, by our hypothesis.)

In particular, let $\delta \in (0,1)$ be a parameter, and let $C$ be a $2^{o(n)}$-size circuit on $n$ inputs as before. We make a circuit $D$ on $m = (1-\delta)n$ inputs with the following behavior: Given an assignment $x$ of $m$ bits, $D$ plugs $x$ into the first $(1-\delta)n$ inputs of $C$, yielding a circuit $C'$ with $\delta n$ inputs and $2^{o(n)}$ size, by calling Circuit Eval appropriately as before. Next, $D$ calls the algorithm $\mathcal{F}_k$ to determine the satisfiability of $C'$ (instead of computing a large OR), which takes $2^{\delta f_k n+o(n)}$ time; converting this call into a circuit, the size is $2^{\delta f_k n+o(n)}$. Now we have a circuit $D$ on $m = (1-\delta)n$ inputs of size $2^{\delta f_k n+o(n)}$ which is equi-satisfiable to our original circuit $C$.

Setting $\delta$ such that $\delta f_k n = \alpha m$, our circuit $D$ will have $m$ inputs and $2^{\alpha m + o(m)}$ size, so its satisfiability can be determined in $2^{\beta m + o(m)}$ time, by our original assumption. Note that

$$\delta f_k n = \alpha m \iff \delta f_k = \alpha(1-\delta),$$

so setting $\delta = \alpha/(f_k + \alpha)$ accomplishes this. We can therefore determine satisfiability of $C$ in time

$$2^{\beta m + o(m)} \le 2^{\beta(1-\alpha/(f_k+\alpha))n + o(n)}.$$

Let this new SAT algorithm be $\mathcal{F}_k$.

Define the sequence

$$f_0 := \beta,\ f_{k+1} := \beta(1 - \alpha/(f_k + \alpha)).$$

The above argument shows that we can construct a sequence of algorithms $\mathcal{F}_k$ for computing satisfiability of $2^{o(n)}$-size circuits, where the $k$th algorithm runs in time $2^{f_k n + o(n)}$. For all $\alpha > 0$, we claim that the sequence $\{f_k\}$ is monotone decreasing, and $\{f_k\}$ converges to

$$f_\infty = \beta - \alpha.$$

First, we note that the sequence $\{f_k\}$ is monotone increasing, by an easy induction proof.

**Base Case:** Showing $f_0 > f_1$ is equivalent to showing $1 > 1 - \alpha/(1+\alpha)$, i.e., $\alpha/(1+\alpha) > 0$, which is true since $\alpha > 0$.

**Inductive Step:** Suppose $f_{k-1} > f_k$. Recall $\beta > \alpha > 0$. We derive

$$\begin{aligned} f_{k+1} = \beta(1 - \alpha/(f_k+\alpha)) < \beta(1 - \alpha/(f_{k-1}+\alpha)) = f_k \\ \Leftrightarrow 1 - \alpha/(f_k+\alpha) < 1 - \alpha/(f_{k-1}+\alpha) \\ \Leftrightarrow \alpha/(f_k+\alpha) > \alpha/(f_{k-1}+\alpha) \\ \Leftrightarrow f_k + \alpha < f_{k-1} + \alpha \\ \Leftrightarrow f_k < f_{k-1}, \text{which we assumed true.} \end{aligned}$$

This completes the induction.

Since every $f_k \ge 0$ and $\{f_k\}$ is monotone decreasing, the sequence has a limit point satisfying the equation

$$f_\infty = \beta(1 - \alpha/(f_\infty + \alpha)),$$

which has the two solutions $f_\infty \in \{0, \beta - \alpha\}$.

The entire construction above is highly uniform, in that a description of the $k$-th algorithm can be constructed in $O(g(k))$ time for a computable function $g$, given that the description length of $S$ is $O(1)$. To ensure that the final running time of our algorithm is indeed $2^{(\beta-\alpha)n+o(n)}$, we can repeat the above construction for a slightly unbounded value $k = k(n)$, and note that the sequence $\{f_k\}$ converges rapidly. We consider two cases. First, for the case where $\alpha = \beta$, one can prove by induction that $f_k = \alpha/(k+\alpha)$. Therefore in this case, for any function $k(n) \ge \omega(1)$, we have $f_{k(n)} \le o(1)$. For the case where $\alpha < \beta$, we have $f_0 - f_1 = \beta\alpha/(1+\alpha)$, and

$$\begin{aligned} f_k - f_{k+1} &= \beta\left(1 - \frac{\alpha}{f_{k-1}+\alpha}\right) - \beta\left(1 - \frac{\alpha}{f_k+\alpha}\right) \\ &= \beta\alpha\left(\frac{1}{f_k+\alpha} - \frac{1}{f_{k-1}+\alpha}\right) \\ &= \beta\alpha\left(\frac{f_{k-1}-f_k}{(f_k+\alpha)(f_{k-1}+\alpha)}\right) \\ &\le \frac{\alpha}{\beta}\cdot(f_{k-1}-f_k), \end{aligned}$$

where the last inequality follows since $f_k + \alpha$ and $f_{k-1} + \alpha$ are both at least $\beta$ (the sequence $\{f_k\}$ is monotone non-decreasing). Therefore for $\beta > \alpha$ and any function $k(n) \ge \omega(1)$, $f_{k(n)}$ is within $o(1)$ of $\beta - \alpha$. This completes the proof. □

By tracking the dependence of the circuit size throughout the proof, one can prove a slightly stronger result than Theorem 1.1, in which the resulting algorithm can solve Circuit SAT rapidly on circuits that are mildly exponential in size.

Theorem 3.1 ([68]). *Suppose there are $\beta \ge \alpha > 0$ such that* Circuit SAT *on $2^{\alpha n + o(n)}$-size circuits can be solved in $2^{\beta n + \varepsilon n}$ time, for all $\varepsilon > 0$. Then for every $\varepsilon > 0$, there is a $\gamma > 0$ such that* Circuit SAT *on $2^{\gamma n}$-size circuits can be solved in $2^{(\beta-\alpha)n+\varepsilon n}$ time.*

## 3.1 Discussion on the Proof

To illustrate the generality of Theorem 1.1, let us discuss various modifications and extensions that can be made. First, note the construction in the proof of Theorem 1.1 works equally well for relating the *circuit complexity* of Circuit SAT and Large Circuit SAT. Replacing every occurrence of "algorithm in time $T$" with "circuit of size $T$" in the proof, every step goes through. We have:

Theorem 3.2. *Let $\alpha, \beta > 0$ with $\alpha \le \beta$. Suppose* Circuit SAT *on $2^{\alpha n + o(n)}$-size circuits can be decided with a circuit family of $2^{\beta n + o(n)}$ size. Then* Circuit SAT *on $2^{o(n)}$-size circuits can be decided with a family of $2^{(\beta-\alpha)n+o(n)}$ size.*

Note that the construction in the proof of Theorem 1.1 is highly *non-black-box*: to solve Circuit SAT on smaller circuits, we use the *descriptions of circuits* solving Circuit SAT in order to form the inputs to other Circuit SAT circuits, and achieve a faster algorithm in each inductive stage. At the same time, there is a sense in which the above proof *relativizes*. Let $A : \{0,1\}^\star \to \{0,1\}$ be an arbitrary oracle, and recall that an *$A$-oracle circuit* is a Boolean circuit equipped with the usual gates, along with copies of $A_k : \{0,1\}^k \to \{0,1\}$, where $A_k$ is the restriction of $A$ to $k$-bit inputs. (Note that because of the unbounded fan-in of the $A_k$ gates, the size of an $A$-oracle circuit is defined to be the number of *wires*, instead of gates.) Given a nontrivially-sized $A$-oracle circuit family for solving Large Circuit SAT on $A$-oracle circuits, the same argument above can be used to derive a smaller $A$-oracle circuit family for Circuit SAT on $A$-oracle circuits of subexponential size.

Theorem 3.3. *Let $\alpha, \beta > 0$ with $\alpha \le \beta$. Suppose* Circuit SAT *on $2^{\alpha n + o(n)}$-size $A$-oracle circuits can be decided by an $A$-oracle circuit family of $2^{\beta n + o(n)}$ size (respectively, an $A$-oracle multitape TM running in $2^{\beta n + o(n)}$ time). Then* Circuit SAT *on $2^{o(n)}$-size $A$-oracle circuits can be decided by an $A$-oracle family of $2^{(\beta-\alpha)n+o(n)}$ size (respectively, an $A$-oracle multitape TM running in $2^{(\beta-\alpha)n+o(n)}$ time).*

It is crucial in our proof that the same oracle $A$ appears in both the instances of Circuit SAT and the algorithmic model solving Circuit SAT: Theorem 1.9 shows that lower bounds such as NP $\neq$ NC$^1$ would follow if we could strengthen self-improvement so that the algorithm can use a stronger oracle than the Circuit SAT instance.

Furthermore, the proof of Theorem 1.1 works with minor modifications for #Circuit SAT, where we wish to *count* the number of SAT assignments to a given circuit, as well as Q-Circuit SAT, the quantified version of Circuit SAT, where we are given a fully-quantified sentence of the form

$$(Q_1\ x_1)\cdots(Q_n\ x_n)[C(x_1,\ldots,x_n)],$$

where each $Q_i \in \{\exists,\forall\}$, $C$ is a circuit, and we wish to decide if the sentence is true or false.

**Reminder of Theorem 1.5.** *Theorem 1.1 holds for* #Circuit SAT *and* Q-Circuit SAT *in place of* Circuit SAT.

Proof. (Sketch) We describe how to modify the proof of Theorem 1.1 to accommodate #Circuit SAT and Q-Circuit SAT.

For Q-Circuit SAT, instead of computing an OR of $2^{m'}$ copies in the base case over all $m'$-bit partial assignments, we compute an appropriate Boolean formula of $2^{m'}$ copies, according to the quantifier types of the $m'$ variables (existential variables get an OR, universal variables get an AND). The remainder of the proof is essentially unchanged: as long as our variable splitting and subsequent calls respect the quantifier order of the variables, the rest of the argument goes through.

For #Circuit SAT, instead of computing an OR of $2^{m'}$ copies in the base case, we instead use a circuit Count which takes $N = 2^{m'}$ bits of input (one for each of the $m'$-bit partial assignments), and outputs the $O(\log N)$-bit count of the number of ones in the input. It is well-known that such a circuit Count can be implemented in $O(N)$ size, and the construction is uniform (see for example [28]). This yields a circuit $D$ which has $m$ inputs, $2^{m'+o(m')}$ size, and $t = O(m')$ outputs. Let the $t$ output bits be numbered $O_{t-1},\ldots,O_0$, so that $O_{t-1}$ is the high-order bit of Count, $O_0$ is the low-order bit, and so on. Define $D_i$ to be the subcircuit of $D$ with only one output gate $O_i$. Then the overall #SAT count of $C$ can be recovered by computing the sum

$$\sum_{i=0}^{t-1} 2^i \cdot \#\text{Circuit SAT}(D_i), \quad (1)$$

which can be done in $\text{poly}(m') \le \text{poly}(n)$ time, given the various #Circuit SAT$(D_i)$. This extra calculation only multiplies the overall running time by $\text{poly}(n) \le 2^{o(n)}$.

To see why (1) is correct, think of the $t$-bit output of the circuit $D$ as an integer in $\{0,1,\ldots,2^t-1\}$, where $D_i$ outputs the $i$-th bit of this integer. We observe:

$$\begin{aligned}\#\text{Circuit SAT}(C) &= \sum_{a\in\{0,1\}^m} D(a) \text{ (by definition of } D)\\ &= \sum_{a\in\{0,1\}^m}\left(\sum_{i=0}^{t-1} 2^i\cdot D_i(a)\right)\\ &\quad\text{(by definition of the circuits } D_i)\\ &= \sum_{i=0}^{t-1} 2^i\cdot\left(\sum_{a\in\{0,1\}^m} D_i(a)\right)\\ &= \sum_{i=0}^{t-1} 2^i\cdot \#\text{Circuit SAT}(D_i).\end{aligned}$$

In the inductive step, the circuit $D$ takes in a partial assignment $x$ of $m$ bits, plugs $x$ into the circuit $C$, then calls an algorithm for #Circuit SAT on the reduced circuit, which then outputs a binary count of the number of satisfying assignments. As in the previous paragraph, we can break $D$ into $t = O(n)$ subcircuits $D_{t-1},\ldots,D_0$ where $D_i$ outputs the $i$-th bit of the binary count. Calling the original assumed #Circuit SAT algorithm on each $D_i(x)$ which has $m$ inputs, we can determine the overall #SAT count using the formula (1). Again, this only multiplies the overall running time by $\text{poly}(n)$ overhead, and computes the exact number of SAT assignments. □

*An Equivalence.* We now turn to proving an equivalence between solving $\varepsilon$-Large Circuit SAT for some $\varepsilon > 0$ and solving $c$-Large Circuit SAT for *every* $c \ge 0$, as mentioned in the Introduction.

**Reminder of Theorem 1.4.** *The following are equivalent:*

1. *There is an $\varepsilon \in (0,1)$ such that $\varepsilon$-*Large Circuit SAT *is in $N^{1+o(1)}$ time.*
2. *For* **every** *$\alpha > 0$ (including arbitrarily large $\alpha$), $\alpha$-*Large Circuit SAT *is in $N^{1+o(1)}$ time.*

Proof. Clearly (2) implies (1). We prove that (1) implies (2). Assume Circuit SAT on $N$ size and $\varepsilon\log(N)$ inputs is in $N^{1+o(1)}$ time for some $\varepsilon > 0$. For every parameter $\alpha > 0$, we want to solve Circuit SAT on $N$ size with $\alpha\log(N)$ inputs. There are two cases:

Suppose $\alpha \le \varepsilon$. Then given a circuit $C$ of $N$ size and $\alpha\log(N)$ inputs, simply add $(\varepsilon-\alpha)\log(N)$ extra "dummy" inputs that do not connect to the rest of $C$. We obtain a circuit $C'$ of size $O(N)$ with $\varepsilon\log(N)$ inputs, and Circuit SAT for $C'$ can be solved in $N^{1+o(1)}$ time.

If $\alpha > \varepsilon$, then let $t$ be the smallest integer such that $\alpha \le t\varepsilon$. Add "dummy" inputs to the circuit $C$ so that $C$ has exactly $t\varepsilon\log(N)$ inputs, and split the inputs of $C$ into $t$ parts of $\varepsilon\log(N)$ variables each.

Set $C_0 := C$. We will show that for all $i = 0,\ldots,t-1$, we can replace our given circuit $C_i$ of size $N^{1+o(1)}$ and $(t-i)\varepsilon\log(N)$ inputs with an equi-satisfiable circuit $C_{i+1}$ that has size $N^{1+o(1)}$ and $(t-(i+1))\varepsilon\log(N)$ inputs. Given the circuit $C_i$ with $(t-i)\varepsilon\log(N)$ inputs, the circuit $C_{i+1}$ will first evaluate $C_i$ on its first $(t-(i+1))\varepsilon\log(N)$ inputs, leaving the last $\varepsilon\log(N)$ inputs free. The resulting circuit description of size $N^{1+o(1)}$ is then fed to the Circuit SAT algorithm for size $N$ and $\varepsilon\log(N)$ inputs, which runs in $N^{1+o(1)}$ time. Converting all the above to circuitry yields a circuit $C_{i+1}$ of $(t-(i+1))\varepsilon\log(N)$ inputs and $(N^{1+o(1)})^{1+o(1)} = N^{1+o(1)}$ size which is equi-satisfiable with $C_i$.

As the final circuit $C_t$ has no inputs and is equi-satisfiable to $C_0 = C$, we obtain an $N^{1+o(1)}$ time algorithm for determining satisfiability of $C$. □

In fact, the equivalence can be strengthened even further, to extensions of satisfiability that correspond to constant levels of the polynomial hierarchy. We naturally define $\Sigma_k$ $\varepsilon$-Large Circuit SAT to be the restriction of Q-Circuit SAT to circuits with $N$ size, $\varepsilon\log(N)$ variables (all quantified), such that it is a "$\Sigma_k$-SAT" instance: namely, the variables can be partitioned into $k$ contiguous blocks, where the first block contains only existentially quantified variables, and for $i = 2,\ldots,k$, block $i$ contains only universally quantified variables if $i$ is even, and existentially quantified variables if $i$ is odd. Observe that $\Sigma_1$ $\varepsilon$-Large Circuit SAT is equivalent to $\varepsilon$-Large Circuit SAT, and $\Sigma_k$ $\alpha$-Large Circuit SAT corresponds to a polynomial-time solvable version of the $\Sigma_k$P-complete problem $\Sigma_k$-SAT [9].

Theorem 3.4 (Extension of Theorem 1.4). *The following are equivalent:*

1. *There is an $\varepsilon \in (0,1)$ such that $\varepsilon$-*Large Circuit SAT *is in $N^{1+o(1)}$ time.*
2. *For* **every** *$\alpha > 0$ (including arbitrarily large $\alpha$), $\alpha$-*Large Circuit SAT *is in $N^{1+o(1)}$ time.*
3. *For every $k \geq 1$ and every $\alpha > 0$, $\Sigma_k$ $\alpha$-*Large Circuit SAT *is in $N^{1+o(1)}$ time.*

Proof. (1) $\iff$ (2) follows from Theorem 1.4. (3) implies (2) by setting $k = 1$.

We prove that (2) implies (3). Given an instance $C$ of the problem $\Sigma_k$Circuit SAT with $\alpha \log(N)$ variables and a circuit predicate of size $N$, first split the variables into $\lceil c/\alpha \rceil$ parts of at most $\alpha \log(N)$ variables each. Next, split every $\alpha \log(N)$-variable part that contains both existential and universal variables ("straddling" multiple quantifier blocks) into smaller parts which only contain variables of the same quantifier type (either all-existential, or all-universal). As there are only $k$ total quantifier blocks, this extra splitting creates at most $k-1$ more parts. Thus the total number of parts $\ell$ is at most $k + \lceil c/\alpha \rceil$, each part has variables of exactly one quantifier type, and each part has at most $\alpha \log(N)$ variables.

Let $C_0 := C$. Applying an analogous argument as in the proof of Theorem 1.4, given a circuit $C_i$ of size $N^{1+o(1)}$ with $\ell - i$ variable parts, in $N^{1+o(1)}$ time we can obtain an equivalent circuit $C_{i+1}$ of size $N^{1+o(1)}$ with $\ell - (i+1)$ variable parts, starting by removing the part that is last in quantification order, and ending with the part that is first in quantification order. Repeating for $\ell - 1$ times, we reduce the $\Sigma_k$Circuit SAT problem to satisfiability on a circuit of size $N^{1+o(1)}$ with $\alpha \log(N)$ variables, which can be determined in $N^{1+o(1)}$ time by assumption. The only remaining issue is how to handle those parts with *universally* quantified variables. Recalling that

$$(\forall x_1, \ldots, x_t)[C(x_1, \ldots, x_t)] \iff \neg(\exists x_1, \ldots, x_t)[\neg C(x_1, \ldots, x_t)],$$

we can decide $(\forall x_1, \ldots, x_t)[C(x_1, \ldots, x_t)]$ by calling

$$\text{Circuit SAT}(\neg C)$$

and flipping the bit of the answer. This amounts to feeding the description of $\neg C_i$ (rather than $C_i$) into our circuit $C_{i+1}$, and flipping the output of the Circuit SAT algorithm implemented in $C_{i+1}$. This completes the proof. ◻

To conclude the discussion, we establish some simple consequences of Theorem 1.1.

**Reminder of Corollary 1.2.** *ETH implies that, for* **every** *$\varepsilon > 0$, $\varepsilon$-*Large Circuit SAT *is not solvable in $N^{1+o(1)}$ time.*

Proof. We prove the contrapositive. Given an instance of Large Circuit SAT with $\varepsilon \log(N)$ inputs and $N$ size, let $n = \varepsilon \log(N)$, so that the circuit size is $N = 2^{\varepsilon n}$. Assuming there is an algorithm running in $N^{1+o(1)} = 2^{\varepsilon n + o(n)}$ time, setting $\alpha = \beta = \varepsilon$, Theorem 1.1 implies that for every $\varepsilon' > 0$, Circuit SAT on $2^{o(n)}$-size circuits can be solved in $2^{\varepsilon' n}$ time. This contradicts (a very weak form of) ETH. ◻

**Reminder of Corollary 1.3.** *Assume for all $\varepsilon > 0$,* Circuit SAT *on $2^{o(n)}$-size circuits cannot be solved in $2^{(1-\varepsilon)n}$ time. Then for every $\alpha \geq 0$ and every $\varepsilon > 0$,* Circuit SAT *on $2^{\alpha n + o(n)}$-size circuits cannot be solved in time $2^{\alpha n + (1-\varepsilon)n + o(n)}$.*

Proof. Again we prove the contrapositive. Suppose there is an $\alpha, \varepsilon > 0$ such that Circuit SAT on $2^{\alpha n + o(n)}$-size circuits has an $2^{\alpha n + (1-\varepsilon)n + o(n)}$-time algorithm. Setting $\beta := \alpha + 1 - \varepsilon$, Theorem 1.1 implies that for every $\varepsilon' > 0$, Circuit SAT on $2^{o(n)}$-size circuits has an $2^{(1-\varepsilon)n + \varepsilon' n + o(n)}$ time algorithm. ◻

## 4 A PARAMETERIZED COMPLEXITY COUNTERPART

Here, we prove Theorem 1.7 from the Introduction. Although the high-level idea of the circuit constructions in this section is similar to that for other self-improvement results in the paper, the analysis and parameter settings turn out to be quite different. For simplicity, we will phrase the results in terms of circuits for Circuit SAT, but (as in other sections) the results hold for any algorithmic model for which time $T$ algorithms can be simulated by size $T^{1+o(1)}$ circuits. Following the notation of parameterized complexity, in the following we let $k$ be the number of input variables to a circuit, and let $n$ be the circuit size.

We start with a lemma showing how to compose two parameterized circuit families for Circuit SAT to obtain a new circuit family:

Lemma 4.1. *Suppose there are $a, c > 0$ and $b, d > 1$ such that for all $n, k \in \mathbb{N}$, there is a $O(2^{ak} + n^b)$-size circuit $A_{n,k}$ and a $O(2^{ck} + n^d)$-size circuit $B_{n,k}$, both solving* Circuit SAT *on instances with $n$ size and $k$ variables. Then there is a circuit family $A'_{n,k}$ solving* Circuit SAT *(with $n$ size and $k$ variables) having size $O(2^{a' \cdot k} + n^{b'})$ for $a' = \frac{adc}{ad+c}$ and $b' = bd$.*

Proof. Our overall approach is similar to the proof of Theorem 1.1: split the set of $k$ inputs into two parts, call the Circuit SAT circuit $A_{n,k}$ on one part forming a new circuit, and call the circuit $B_{n,k}$ for Circuit SAT on the new circuit obtained.

Let $\rho \in (0,1)$ be a parameter to be set later. Given a circuit $C(x, y)$ of size $n$ (represented in $O(n \log n)$ bits) with $\rho k$ inputs $x$ and $(1-\rho)k$ inputs $y$, we construct a new circuit $C'$ defined as follows:

$$C'(y) := A_{n,\rho k}(C(x, y)).$$

That is, $C'$ has $(1-\rho)k$ free inputs of $C$; for any such assignment to those inputs, $C'$ calls $A_{n,\rho k}$ on the resulting circuit of size at most $n$ with $\rho k$ inputs. By assumption, the size of the new circuit $C'$ is

$$|C'| \leq O(n \log n + 2^{a\rho k} + n^b).$$

Note that $C'$ is satisfiable if and only if $C$ is satisfiable. To solve satisfiability for $C'$, we can call $B_{n',(1-\rho)k}$ on $C'$, where $n'$ is the size of $C'$. The resulting composition of $A_{n,\rho k}$ and $B_{n',(1-\rho)k}$ yields a circuit for satisfiability of $n$-size $k$-input circuits, which has size

$$\begin{aligned} O(2^{c(1-\rho)k} &+ (n \cdot \text{poly}(\log n) + 2^{a\rho k} + n^b)^d) \\ &\leq O(2^{c(1-\rho)k} + 2^{ad\rho k} + n^{bd}). \end{aligned}$$

To minimize the dependence on $k$, we set $\rho = \frac{c}{ad+c}$. The size is then $O(2^{a'k} + n^{b'})$, where $a' = \frac{adc}{ad+c}$ and $b' = bd$. □

A key observation is that, when $a = c$ in the above, the new exponent $a' = a \cdot \frac{d}{d+1} < a$. Indeed, this (strict) inequality is true for any $a \geq c$. Therefore, applying a Circuit SAT circuit family of size $2^{ak} + n^b$ to *itself* in lemma 4.1 yields a decrease in the exponent with respect to $k$. We will exploit this fact.

Let us begin with a simpler statement which is easier to prove, but still conveys the main idea:

Theorem 4.2 (Theorem 1.7, Simplified). *There is a $c > 1$ such that* Circuit SAT *has circuits of size $O(c^k) + n^{1+o(1)}$, if and only if* **for every** $c > 1$, Circuit SAT *has circuits of size $O(c^k) + n^{1+o(1)}$.*

Proof. One direction is immediate: the statement with a universal quantifier on $c$ clearly implies the statement with an existential quantifier on $c$. For the other direction, assume we start with circuits of $O(2^{ak}) + n^{1+o(1)}$ size for Circuit SAT for some constant $a \geq 1$. For a large constant $t$, we will apply Lemma 4.1 for $t$ times to these circuits. In particular, we start by using the assumed circuits in place of $A_{n,\rho k}$, and in place of $B_{n',(1-\rho)k}$ in Lemma 4.1. In each subsequent application of Lemma 4.1, we take the Circuit SAT circuits obtained from the previous application, and again apply those circuits *twice* in Lemma 4.1 to form new circuits (we use the ciircuits from the previous application as circuit $A_{n,\rho k}$, and as circuit $B_{n',(1-\rho)k}$). Since in our case we always have $b = 1 + o(1)$ and $d = 1 + o(1)$, when we apply circuits of size $O(2^{ck}) + n^{1+o(1)}$ to circuits of size $O(2^{ak}) + n^{1+o(1)}$ in Lemma 4.1, we obtain circuits of size

$$2^{a'k+o(k)} + n^{1+o(1)},$$

where $a' = ac/(a+c)$. (Here we are using the fact that $n^{(1+o(1))^2} \leq n^{1+o(1)}$.) Since we are always applying the same Circuit SAT circuit *twice* to obtain the next circuit, we also have $a = c$. Therefore after one application of Lemma 4.1, we have

$$a' = a^2/(2a) = a/2,$$

square-rooting the exponential dependence on the number of variables $k$. After $t$ applications, it follows by induction that the circuit size is at most $2^{ak/2^t+o(k)} + n^{1+o(1)}$. Therefore we can set $t$ to be an arbitrarily large constant, and obtain circuits of $2^{\alpha k} + n^{1+o(1)}$ size for Circuit SAT, for any desired $\alpha > 0$. □

Now we move to the version of Theorem 1.7 presented in the introduction, in which a slightly super-linear dependence on $n$ is allowed; the proof turns out to be more delicate.

Theorem 4.3 (Theorem 1.7, Rephrased in terms of circuits). *There is a $c > 1$ such that for all $\varepsilon > 0$,* Circuit SAT *has $O(c^k+n^{1+\varepsilon})$-size circuits, if and only if* **for every** $c > 1$ *and $\varepsilon > 0$,* Circuit SAT *has $O(c^k + n^{1+\varepsilon})$-size circuits.*

Proof. One direction is immediate: the statement with a universal quantifier on $c$ clearly implies the statement with an existential quantifier on $c$. For the other direction, we show:

> If there is an $a \geq 1$ so that for all $k, n$ there are circuits $A_{n,k}$ solving Circuit SAT (on instances with $k$ inputs and $n$ size) of size $O(2^{ak} + n^{1+\varepsilon})$ for all $\varepsilon > 0$, then for all $\alpha, \beta > 0$ there are $O(2^{\alpha k} + n^{1+\beta})$-size circuits $A'_{n,k}$ solving Circuit SAT (on $k$ inputs and $n$ size).

Let $\alpha, \beta > 0$ be arbitrarily small, and let $a \geq 1$ be given. Let $b = 1+\varepsilon$ for an arbitrarily small $\varepsilon > 0$ that we will set later, so that our assumed Circuit SAT circuits have size $O(2^{\alpha k} + n^b)$.

Let $t \geq 1$ be an integer parameter. Following the proof of Theorem 4.2, we apply Lemma 4.1 for $t$ times to our assumed Circuit SAT circuit, where each time we use the Circuit SAT circuits previously obtained *twice*: as the circuit $A_{n,k}$ and as the circuit $B_{n,k}$.

Let us analyze the effect of these Lemma 4.1 applications on the exponent pairs $a, b$. At the start, we have $a_1 = a$ and $b_1 = b$. After the $(i+1)$-th application of Lemma 4.1, we obtain circuits of size $O(2^{a_{i+1}\cdot k} + n^{b_{i+1}})$, where

$$a_{i+1} = \frac{(a_i)^2 \cdot b_i}{a_i \cdot b_i + a_i} = a_i \cdot \frac{b_i}{1+b_i},$$

and

$$b_{i+1} = (b_i)^2.$$

Inductively, we obtain $b_{i+1} = b^{2^i}$ and

$$a_{i+1} = a \cdot \prod_{j=0}^{i-1} \left( \frac{b^{2^j}}{1+b^{2^j}} \right).$$

Recall that $b = 1 + \varepsilon$. We want to show that $\varepsilon > 0$ and $t \geq 1$ can be set in such a way that two inequalities hold simultaneously:

$$b_{t+1} = (1+\varepsilon)^{2^t} \leq 1 + \beta, \tag{2}$$

and

$$a_{t+1} = a \cdot \prod_{j=0}^{t-1} \left( \frac{b^{2^j}}{1+b^{2^j}} \right) \leq \alpha. \tag{3}$$

For now, let us suppose that after the parameter $t$ is determined, $\varepsilon > 0$ is always set small enough that $(1+\varepsilon)^{2^t} = 1 + \beta$, satisfying inequality (2). That is, we think of $\varepsilon$ as a function of $t$: whatever $t$ is set to, $\varepsilon$ will be set accordingly. Now we can focus on satisfying inequality (3). First, we rewrite (3) so that it reads:

$$a_{t+1} = a \cdot \prod_{j=0}^{t-1} \left( 1 - \frac{1}{1+b^{2^j}} \right) \leq \alpha.$$

Consider applying the inequality $1 - x \leq e^{-x}$ (for $x \geq 0$) to each term of the above product. Inequality (3) would then be satisfied, if the following inequality is true:

$$a \cdot e^{-\sum_{j=0}^{t-1} \frac{1}{1+b^{2^j}}} \leq \alpha. \tag{4}$$

Focusing on the exponent in inequality (4), we see that

$$\sum_{j=0}^{t-1} \frac{1}{1+b^{2^j}} \geq \frac{t}{1+b^{2^t}} = \frac{t}{1+(1+\varepsilon)^{2^t}},$$

since each of the $t$ terms is lower bounded by $1/(1+(1+\varepsilon)^{2^t})$. Recall that we have resolved to set $\varepsilon > 0$ so that $(1+\varepsilon)^{2^t} = 1+\beta$. Therefore, if $t$ is set so that

$$a \cdot e^{-\frac{t}{2+\beta}} \leq \alpha, \tag{5}$$

we would satisfy inequality (4), and thereby inequality (3). But inequality (5) is true as long as

$$t \geq (2+\beta)\ln(a/\alpha).$$

Recall that $a, \alpha, \beta$ were all fixed constants. Therefore, by setting $t$ large enough, and then setting $\varepsilon > 0$ small enough that $(1+\varepsilon)^{2^t} = 1+\beta$, we have obtained a circuit family of size $O(2^{\alpha k} + n^{1+\beta})$. □

One can also derive analogous "parameterized self-improvement" theorems for #Circuit SAT and Q-Circuit SAT; we omit the details, but they can be interpolated from the other proofs in this paper.

## 5 OPEN PROBLEMS

We conclude with a few intriguing open problems.

**Could self-improvement go all the way down to P = NP?** Is it possible that (say) linear-time SAT algorithms for exponential-size circuits might imply polynomial-time algorithms for polynomial-size circuits, concluding P = NP? There seem to be bottlenecks in the current argument that prevent us from going significantly below subexponential time, but they could possibly be circumvented with a little cleverness. Could self-improvement be strengthened in the non-uniform case to conclude NP ⊂ P/poly?

**Could self-improvement-style results hold for other combinatorial problems, besides just circuit-based ones?** On the face of it, it seems crucial in our self-improvement results that the algorithm solving the problem can be modeled extremely efficiently, within an instance of the problem. But given the ubiquity of complete problems for NP, PSPACE, and so on, it seems possible that self-improvement phenomena could arise in many domains. To give one example of where something like self-improvement may arise in graph algorithms, an insightful paper of Or Zamir [70] uses the container method to show (among other results) that if maximum independent set (MIS) can be solved in $c^n$ time on $n$-node graphs for some $c > 1$, then it can be solved even faster on $d$-regular graphs, in $c^{n/2+o_d(n)}$ time. It follows from this reduction that, if there were a fine-grained reduction (running in subexponential time, preserving the parameter $n$) from the general case of MIS to the $d$-regular case for large enough constant $d$, then ETH would be false: we could repeatedly alternate between the hypothetical fine-grained reduction and Zamir's reduction, reducing the running time exponent for MIS to be as small as we liked.

**Can the uniform SYM∘SYM circuit lower bounds be further improved?** In principle, some fast matrix multiplication algorithms can be implemented in $\mathsf{TC}^0$ [51] so one might hope to reduce the complexity of the hard function further in our lower bound. There may also be a way to improve the degree of the polynomial in the lower bound, by applying self-improvement. Finally, it seems plausible that our lower bound might be extended to prove that, for every $d$, there is a $c_d > 1$ such that Circuit Eval does not have depth-$d$ SYM circuits of $O(n^{c_d})$ gates.

*Acknowledgements.* I am grateful to Lijie Chen, Russell Impagliazzo, Valentine Kabanets, Mohan Paturi, Rahul Santhanam, and Michael Wehar for interesting and useful discussions. I also thank Shyan Akmal and the STOC'24 reviewers for comments and corrections on an earlier version of the manuscript. Some results in the parameterized complexity setting were originally obtained in collaboration with Nikhil Vyas, but he (sadly) declined co-authorship on this paper. Many thanks to the Simons Institute for providing a stimulating environment during the Meta-Complexity and Satisfiability Extended Reunion programs in Spring 2023.

This paper is dedicated to the memory of my undergraduate mentor Juris Hartmanis, who posed research questions of the form that are addressed in this paper (see also [4]). Namely, Prof. Hartmanis often asked me: "If P = NP, then can SAT be solved in $n^{10}$ time?" (One may substitute "10" with any specific constant.) The present paper is my current best attempt to prove that some "fine-grained" version of his question can be answered positively.

## REFERENCES


[1] Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. 2016. Simulating branching programs with edit distance and friends: or: a polylog shaved is a lower bound made. In *STOC*. ACM, 375–388. https://doi.org/10.1145/2897518.2897653

[2] Amir Abboud, Ryan Williams, and Huacheng Yu. 2015. More applications of the polynomial method to algorithm design. In *SODA*. 218–230.

[3] Manindra Agrawal, Sumanta Ghosh, and Nitin Saxena. 2018. Bootstrapping variables in algebraic circuits. In *STOC*. ACM, 1166–1179. https://doi.org/10.1145/3188745.3188762

[4] Eric Allender, Jin-Yi Cai, Lance Fortnow, William Gasarch, Neil Immerman, Stuart Kurtz, James Royer, and Ryan Williams. 2022. Open Problems Column: Open Problems by or Inspired by Juris Hartmanis. *SIGACT News* 53, 4 (2022), 26.

[5] Eric Allender and Vivek Gore. 1994. A Uniform Circuit Lower Bound for the Permanent. *SIAM J. Comput.* 23, 5 (1994), 1026–1049. https://doi.org/10.1137/S0097539792233907

[6] Eric Allender and Michal Koucký. 2010. Amplifying lower bounds by means of self-reducibility. *J. ACM* 57, 3 (2010), 14:1–14:36.

[7] Eric Allender, Michal Koucký, Detlef Ronneburger, Sambuddha Roy, and V. Vinay. 2001. Time-Space Tradeoffs in the Counting Hierarchy. In *CCC*. 295–302. https://doi.org/10.1109/CCC.2001.933896

[8] Josh Alman, Timothy M. Chan, and R. Ryan Williams. 2016. Polynomial Representations of Threshold Functions and Algorithmic Applications. In *FOCS*. 467–476.

[9] Sanjeev Arora and Boaz Barak. 2009. *Computational Complexity - A Modern Approach.* Cambridge University Press.

[10] Vishwas Bhargava, Sumanta Ghosh, Zeyu Guo, Mrinal Kumar, and Chris Umans. 2022. Fast Multivariate Multipoint Evaluation Over All Finite Fields. In *FOCS*. IEEE, 221–232.

[11] Vishwas Bhargava, Sumanta Ghosh, Mrinal Kumar, and Chandra Kanta Mohapatra. 2022. Fast, algebraic multivariate multipoint evaluation in small characteristic and applications. In *STOC*. ACM, 403–415.

[12] Stephen A. Bloch, Jonathan F. Buss, and Judy Goldsmith. 1998. Sharply Bounded Alternation and Quasilinear Time. *Theory Comput. Syst.* 31, 2 (1998), 187–214. https://doi.org/10.1007/s002240000085

[13] Allan Borodin and R. Moenck. 1974. Fast Modular Transforms. *J. Comput. Syst. Sci.* 8, 3 (1974), 366–386. https://doi.org/10.1016/S0022-0000(74)80029-2

[14] Mark Braverman, Young Kun-Ko, and Omri Weinstein. 2015. Approximating the best Nash Equilibrium in $n^{o(\log n)}$-time breaks the Exponential Time Hypothesis. In *SODA*. 970–982. https://doi.org/10.1137/1.9781611973730.66

[15] Jonathan F. Buss and Judy Goldsmith. 1993. Nondeterminism Within P. *SIAM J. Comput.* 22, 3 (1993), 560–572. https://doi.org/10.1137/0222038

[16] Samuel R. Buss and Ryan Williams. 2015. Limits on Alternation Trading Proofs for Time-Space Lower Bounds. *Comput. Complex.* 24, 3 (2015), 533–600. https://doi.org/10.1007/s00037-015-0104-9

[17] Liming Cai and Jianer Chen. 1997. On the Amount of Nondeterminism and the Power of Verifying. *SIAM J. Comput.* 26, 3 (1997), 733–750. https://doi.org/10.1137/S0097539793258295

[18] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. 2006. A Duality between Clause Width and Clause Density for SAT. In *CCC*. 252–260.

[19] Timothy M. Chan and R. Ryan Williams. 2021. Deterministic APSP, Orthogonal Vectors, and More: Quickly Derandomizing Razborov-Smolensky. *ACM Trans. Algorithms* 17, 1 (2021), 2:1–2:14. https://doi.org/10.1145/3402926

[20] Lijie Chen, Shuichi Hirahara, Igor Carboni Oliveira, Ján Pich, Ninad Rajgopal, and Rahul Santhanam. 2020. Beyond Natural Proofs: Hardness Magnification and Locality. In *ITCS*. 70:1–70:48. https://doi.org/10.4230/LIPIcs.ITCS.2020.70

[21] Lijie Chen, Ce Jin, and R. Ryan Williams. 2019. Hardness Magnification for all Sparse NP Languages. In *FOCS*. 1240–1255.

[22] Lijie Chen, Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. 2019. Relations and Equivalences Between Circuit Lower Bounds and Karp-Lipton Theorems. In *CCC*. 30:1–30:21. https://doi.org/10.4230/LIPIcs.CCC.2019.30
[23] Lijie Chen, Ron D. Rothblum, Roei Tell, and Eylon Yogev. 2020. On Exponential-Time Hypotheses, Derandomization, and Circuit Lower Bounds: Extended Abstract. In *FOCS*. IEEE, 13–23.
[24] James W Cooley and John W Tukey. 1965. An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation* 19, 90 (1965), 297–301.
[25] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. 2015. *Parameterized Algorithms*. Springer. https://doi.org/10.1007/978-3-319-21275-3
[26] Marek Cygan, Marcin Pilipczuk, and Michal Pilipczuk. 2016. Known Algorithms for Edge Clique Cover are Probably Optimal. *SIAM J. Comput.* 45, 1 (2016), 67–83. https://doi.org/10.1137/130947076
[27] Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslaman, and Martin Wahlen. 2014. Exponential Time Complexity of the Permanent and the Tutte Polynomial. *ACM Trans. Algorithms* 10, 4 (2014), 21:1–21:32. https://doi.org/10.1145/2635812
[28] Evgeny Demenkov, Arist Kojevnikov, Alexander S. Kulikov, and Grigory Yaroslavtsev. 2010. New upper bounds on the Boolean circuit complexity of symmetric functions. *Inf. Process. Lett.* 110, 7 (2010), 264–267. https://doi.org/10.1016/j.ipl.2010.01.007
[29] Rodney G. Downey and Michael R. Fellows. 1999. *Parameterized Complexity*. Springer. https://doi.org/10.1007/978-1-4612-0515-9
[30] Charles M. Fiduccia. 1972. Polynomial Evaluation via the Division Algorithm: The Fast Fourier Transform Revisited. In *STOC*. 88–93.
[31] Jörg Flum, Martin Grohe, and Mark Weyer. 2006. Bounded fixed-parameter tractability and $\log^2 n$ nondeterministic bits. *J. Comput. Syst. Sci.* 72, 1 (2006), 34–71. https://doi.org/10.1016/j.jcss.2005.06.003
[32] Lance Fortnow, Richard J. Lipton, Dieter van Melkebeek, and Anastasios Viglas. 2005. Time-space lower bounds for satisfiability. *J. ACM* 52, 6 (2005), 835–865.
[33] Sumanta Ghosh, Prahladh Harsha, Simao Herdade, Mrinal Kumar, and Ramprasad Saptharishi. 2023. Fast Numerical Multivariate Multipoint Evaluation. *Electronic Colloquium on Computational Complexity* TR23-033 (2023).
[34] Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. 2008. Data reduction and exact algorithms for clique cover. *ACM J. Exp. Algorithmics* 13 (2008). https://doi.org/10.1145/1412228.1412236
[35] Russell Impagliazzo and Ramamohan Paturi. 1999. Complexity of $k$-SAT. In *CCC*. 237–240. https://doi.org/10.1109/CCC.1999.766282
[36] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. 2001. Which Problems Have Strongly Exponential Complexity? *J. Comput. Syst. Sci.* 63, 4 (2001), 512–530. https://doi.org/10.1006/jcss.2001.1774
[37] Stasys Jukna. 2012. *Boolean function complexity: advances and frontiers*. Vol. 27. Springer Science & Business Media.
[38] Daniel M. Kane and Ryan Williams. 2016. Super-linear gate and super-quadratic wire lower bounds for depth-two and depth-three threshold circuits. In *STOC*. 633–643.
[39] Ravi Kannan. 1983. Alternation and the Power of Nondeterminism. In *STOC*. ACM, 344–346.
[40] Kiran S. Kedlaya and Christopher Umans. 2011. Fast Polynomial Factorization and Modular Composition. *SIAM J. Comput.* 40, 6 (2011), 1767–1802.
[41] Chandra M. R. Kintala and Patrick C. Fischer. 1977. Computations with a Restricted Number of Nondeterministic Steps (Extended Abstract). In *STOC*. ACM, 178–185. https://doi.org/10.1145/800105.803407
[42] Mrinal Kumar, Ramprasad Saptharishi, and Anamay Tengse. 2019. Near-optimal Bootstrapping of Hitting Sets for Algebraic Circuits. In *SODA*. SIAM, 639–646. https://doi.org/10.1137/1.9781611975482.40
[43] Mrinal Kumar, Ramprasad Saptharishi, and Anamay Tengse. 2023. Near-Optimal Bootstrapping of Hitting Sets for Algebraic Models. *Theory of Computing* 19, 12 (2023), 1–30. https://doi.org/10.4086/toc.2023.v019a012
[44] Francois Le Gall and Florent Urrutia. 2018. Improved Rectangular Matrix Multiplication using Powers of the Coppersmith-Winograd Tensor. In *SODA*. SIAM, 1029–1046.
[45] Richard J. Lipton, Evangelos Markakis, and Aranyak Mehta. 2003. Playing large games using simple strategies. In *Proceedings 4th ACM Conference on Electronic Commerce (EC)*. ACM, 36–41. https://doi.org/10.1145/779928.779933
[46] Richard J. Lipton and Ryan Williams. 2013. Amplifying circuit lower bounds against polynomial time, with applications. *Computational Complexity* 22, 2 (2013), 311–343. https://doi.org/10.1007/s00037-013-0069-5
[47] Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. 2019. Weak lower bounds on resource-bounded compression imply strong separations of complexity classes. In *STOC*. ACM, 1215–1225.
[48] Abhijit Mudigonda and R. Ryan Williams. 2021. Time-Space Lower Bounds for Simulating Proof Systems with Quantum and Randomized Verifiers. In *ITCS (LIPIcs, Vol. 185)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 50:1–50:20.
[49] Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam. 2019. Hardness Magnification near State-Of-The-Art Lower Bounds. In *34th Computational Complexity Conference, CCC 2019*. 27:1–27:29. https://doi.org/10.4230/LIPIcs.CCC.2019.27
[50] Igor Carboni Oliveira and Rahul Santhanam. 2018. Hardness Magnification for Natural Problems. In *FOCS*. 65–76.
[51] Ojas Parekh, Cynthia A. Phillips, Conrad D. James, and James B. Aimone. 2018. Constant-Depth and Subcubic-Size Threshold Circuits for Matrix Multiplication. In *SPAA*. ACM, 67–76. https://doi.org/10.1145/3210377.3210410
[52] Ramamohan Paturi and Pavel Pudlák. 2010. On the complexity of circuit satisfiability. In *STOC*. ACM, 241–250.
[53] Wolfgang J. Paul. 1976. Realizing Boolean Functions on Disjoint sets of Variables. *Theor. Comput. Sci.* 2, 3 (1976), 383–396. https://doi.org/10.1016/0304-3975(76)90089-X
[54] Nicholas Pippenger and Michael J. Fischer. 1979. Relations among complexity measures. *J. ACM* 26, 2 (1979), 361–381.
[55] Aviad Rubinstein. 2016. Settling the Complexity of Computing Approximate Two-Player Nash Equilibria. In *FOCS*. 258–265.
[56] András Z. Salamon and Michael Wehar. 2022. Superlinear Lower Bounds Based on ETH. In *STACS (LIPIcs, Vol. 219)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 55:1–55:16. https://doi.org/10.4230/LIPIcs.STACS.2022.55
[57] Rahul Santhanam. 2023. An Algorithmic Approach to Uniform Lower Bounds. *Electronic Colloquium on Computational Complexity (ECCC)* TR23-028 (2023), 100.
[58] Aravind Srinivasan. 2003. On the approximability of clique and related maximization problems. *J. Comput. Syst. Sci.* 67, 3 (2003), 633–651. https://doi.org/10.1016/S0022-0000(03)00110-7
[59] Suguru Tamaki. 2016. A Satisfiability Algorithm for Depth Two Circuits with a Sub-Quadratic Number of Symmetric and Threshold Gates. *Electronic Colloquium on Computational Complexity (ECCC)* TR16-100 (2016). http://eccc.hpi-web.de/report/2016/100
[60] Leslie G. Valiant. 1976. Universal Circuits (Preliminary Report). In *STOC*. ACM, 196–203.
[61] Dieter van Melkebeek. 2006. A Survey of Lower Bounds for Satisfiability and Related Problems. *Found. Trends Theor. Comput. Sci.* 2, 3 (2006), 197–303. https://doi.org/10.1561/0400000012
[62] Dieter van Melkebeek and Ran Raz. 2005. A time lower bound for satisfiability. *Theor. Comput. Sci.* 348, 2-3 (2005), 311–320. https://doi.org/10.1016/j.tcs.2005.09.020
[63] Dieter van Melkebeek and Thomas Watson. 2012. Time-Space Efficient Simulations of Quantum Computations. *Theory Comput.* 8, 1 (2012), 1–51. https://doi.org/10.4086/toc.2012.v008a001
[64] Virginia Vassilevska Williams. 2018. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians (ICM)*. World Scientific, 3447–3487.
[65] Heribert Vollmer. 1999. *Introduction to Circuit Complexity - A Uniform Approach*. Springer. https://doi.org/10.1007/978-3-662-03927-4
[66] Ryan Williams. 2013. Improving Exhaustive Search Implies Superpolynomial Lower Bounds. *SIAM J. Comput.* 42, 3 (2013), 1218–1244. https://doi.org/10.1137/10080703X
[67] Ryan Williams. 2014. Nonuniform ACC circuit lower bounds. *J. ACM* 61, 1 (2014), 2.
[68] Ryan Williams. 2023. Self-Improvement for Circuit-Analysis Problems. *Electron. Colloquium Comput. Complex.* TR23-082 (2023). ECCC:TR23-082 https://eccc.weizmann.ac.il/report/2023/082
[69] R. Ryan Williams. 2018. New Algorithms and Lower Bounds for Circuits With Linear Threshold Gates. *Theory of Computing* 14, 1 (2018), 1–25.
[70] Or Zamir. 2023. Algorithmic Applications of Hypergraph and Partition Containers. In *STOC*. ACM, 985–998. https://doi.org/10.1145/3564246.3585163

Received 12-NOV-2023; accepted 2024-02-11