Electronic Colloquium on Computational Complexity, Report No. 82 (2023)

Self-Improvement for Circuit-Analysis Problems

Ryan Williams*
MIT

Dedicated to the memory of Juris Hartmanis

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. 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 how slightly faster #Circuit-SAT algorithms on large circuits can be used to prove lower bounds against uniform circuits with symmetric gates for functions in deterministic linear time. Our result suggests an "algorithmic method" approach for uniform circuit lower bounds, which trades non-uniformity for a substantial reduction in the complexity of the hard function.

*This work was supported in part by the Simons Institute at UC Berkeley, NSF CCF-2127597, and a Frank Quick Faculty Research Innovation Fellowship. Email: rrw@mit.edu.

ISSN 1433-8092

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, [CPP16] prove that the EDGE CLIQUE COVER problem, which has a simple $2^{2^k} \cdot \text{poly}(n)$ time algorithm [GGHN08], 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 [LMM03], it is also known [BKW15] that an $n^{o(\log n)}$-time approximation algorithm (with "good social welfare") would contradict ETH (see also [Rub16]).

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 [Vas18] 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 [IPZ01, CIP06]. Conversely, the question of whether ETH implies SETH is a major open problem (already raised explicitly in [IP99]). It is entirely uncertain how such an implication might be proved. In this paper, we ask a more general question:

> **Question:** *Is it possible to "amplify" weak exponent lower bounds 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 the CIRCUIT SAT problem, as well as its counting and quantified variants.

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 $\varepsilon \log(N)$ inputs and N gates (a.k.a. N size).[1]
Decide: Is there an $a \in \{0,1\}^n$ such that $C(a) = 1$?

[1] The circuits can be over any universal basis of constant fain, e.g., AND/OR/NOT.

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 $o(N^{\varepsilon})$ [Juk12]. 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 ε), as one can reduce from the "large" circuit case to the "small" circuit case by simply adding dummy inputs (see Theorem 3.4 for one formalization of this intuition). Observe the brute-force algorithm for LARGE CIRCUIT SAT takes nearly-linear time, about $\tilde{O}(N^{1+\varepsilon})$ 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* $\varepsilon > 0$?

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 main theorem is as follows.

Theorem 1.1 ("Self-Improvement" For Circuit-SAT, Section 3)**.** *Let α, β 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.*

We call such a result *self-improving*, as it proceeds by induction, and 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 [PF79]).[2] Theorem 1.1 also holds for randomized algorithms (see Appendix B) 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 1.1 (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. Furthermore, we can prove an equivalence between nearly-linear-time algorithms for CIRCUIT SAT on $\varepsilon \log(N)$ inputs for arbitrarily small $\varepsilon > 0$, CIRCUIT SAT on $K \log(N)$ inputs for arbitrarily *large* $K \geq 1$, and extensions of CIRCUIT SAT that correspond to levels of the polynomial hierarchy (Theorem 3.4 and Theorem 3.5). It is also instructive to compare Corollary 1.1 with the implications obtained by assuming a CIRCUIT SAT form of SETH, rather than ETH:

Corollary 1.2 (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 $2^{\alpha n+(1-\varepsilon)n+o(n)}$.*

[2] We discuss in Section 3.2 how to obtain results for (apparently) 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.

That is, if brute-force 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 3.4). Phrasing Corollary 1.2 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.

We now describe various extensions and consequences. One consequence is that we can prove an *equivalence* between fast CIRCUIT SAT algorithms for the large-exponential-size and small-exponential-size cases, as well as an equivalence with Σ_kCIRCUIT SAT for constant $k \geq 1$ (see Theorem 3.4 for details).

#SAT and QBF. The proof of Theorem 1.1 is 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 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 1.2. *Theorem 1.1 holds for* #CIRCUIT SAT *and* Q-CIRCUIT SAT *in place of* CIRCUIT SAT.

Thus, all the corollaries for strong-exponent and weak-exponent lower bounds for CIRCUIT SAT carry over for #CIRCUIT SAT and Q-CIRCUIT SAT as well.

An FFT for Circuits Would Refute Exponential-Time Hypotheses. A major application of the Fast Fourier Transform (FFT) [CT65] is that univariate degree-n polynomials over a field can be evaluated on any n points in $n \cdot \text{poly}(\log n)$ operations [Fid72, BM74], a great improvement over the obvious $\Theta(n^2)$ algorithm. Recent work has extended this fundamental result to the multivariate setting [KU11, BGKM22, BGG+22, GHH+23].

Should we expect fast multipoint evaluation for more complex computational models, such as Boolean circuits? On the one hand, a folklore result[3] 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 1.2 is that significantly weaker hypotheses suffice:

Corollary 1.3. *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 [DHM+14] and the quantified version of ETH [CRTY20] are false:* #CIRCUIT SAT *and* Q-CIRCUIT SAT *on n-input circuits of $2^{o(n)}$ size can be solved 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 far weaker hypotheses than SETH: weaker than even circuit versions of SETH (used to argue for the hardness of problems like Edit Distance [AHWW16]).

[3] See https://cstheory.stackexchange.com/questions/3085/are-there-known-to-exist-functions-with-the-following-direct-sum-property

[4] For example, see footnote 7 in [Wil13].

A Uniform Circuit Lower Bound for Linear Time. Studying the consequences of faster #CIRCUIT SAT algorithms for large circuits further, we show how to 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 E^{NP} which do not have non-uniform $\mathsf{SYM} \circ \mathsf{SYM}$ circuits of $n^{2-\varepsilon}$ gates, for all $\varepsilon > 0$ [ACW16, Tam16]. Since $\mathsf{SYM} \circ \mathsf{SYM}$ can be simulated in depth-3 TC^0 with a polynomial blowup in size, one can deduce from known results on TC^0 ([AG94]) 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 [AK10].[5] We prove a super-linear gate lower bound for computing problems in linear time with uniform $\mathsf{SYM} \circ \mathsf{SYM}$ circuits.

Theorem 1.3 (Section 4)**.** *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 for example take the CIRCUIT EVALUATION decision problem.) The proof of Theorem 1.3 has the form of an indirect diagonalization: assuming the opposite, we derive a simulation of time-bounded computation that contradicts a hierarchy theorem. However, for all prior indirect diagonalization lower bounds that we are aware of, across a variety of models (such as [Kan83, AG94, AKR$^+$01, FLvMV05, vMR05, vM06, vMW12, MW21]), the proofs require that the hard function is *much* harder than linear-time computable. For example, the time-space tradeoffs for SAT [FLvMV05, vM06, BW15] crucially require that the hard function is NP-hard under extremely local reductions.

We deduce a contradiction by exploiting *circuit-analysis algorithms* for $\mathsf{SYM} \circ \mathsf{SYM}$. That is, we establish a version of the "algorithmic method" for circuit lower bounds (initiated by Williams [Wil13, Wil14]) that applies to *uniform* circuits, and allows the hard function to be contained in P.[6] 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 [LU18] appropriately to "speed-up" the evaluation of a $\mathsf{SYM} \circ \mathsf{SYM}$ circuit. If the matrix multiplication exponent ω happens to be 2, our gate lower bound would improve to $n^{1.36}$.

Indeed, matrix multiplication allows us to compute truth tables of $\mathsf{SYM} \circ \mathsf{SYM}$ 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 [Wil18], who gave a faster truth-table evaluation algorithm for $\mathsf{THR} \circ \mathsf{THR}$ circuits, and asked if such algorithms suffice for deriving lower bounds. (However, super-linear gate lower bounds against $\mathsf{THR} \circ \mathsf{THR}$ are already known; see [KW16]. Thus we instead state our results in terms of $\mathsf{SYM} \circ \mathsf{SYM}$.) 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 E^{NP} or QuasiNP, down to linear time).

[5] Note that, although it is also known [KW16] that there are functions in P that require $n^{1.5-o(1)}$ gates to be computed by non-uniform depth-3 TC^0 circuits, the translation of $\mathsf{SYM} \circ \mathsf{SYM}$ into depth-3 TC^0 can increase the total number of gates by a factor of n, so the methods of [KW16] do not seem to yield linear gate lower bounds for $\mathsf{SYM} \circ \mathsf{SYM}$ circuits.

[6] See [San23] 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.

On the Difficulty of Improving Self-Improvement. Our results yield a new perspective on computational lower bounds. We already mentioned (Theorem 1.2) that self-improvement holds for the Q-CIRCUIT SAT (Quantified Circuit SAT) problem. First we 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 α,* 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 Σ_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 $O(2^{kn} + 2^n)$ time. (We can use $\Sigma_2 SAT$ as an oracle specifically, because of tight reductions from Σ_2 time $T(n)$ to $\Sigma_2 SAT$; see for example [FLvMV05].) This observation naturally begs the question of whether Q-CIRCUIT SAT self-improvement is possible on algorithms with an oracle in $\Sigma_2\mathsf{P}$. We observe that such a result would separate NP from NC^1, even if we could only obtain non-uniform circuits as a consequence.

Theorem 1.4 (Section 5)**.** *Suppose self-improvement holds for* Q-CIRCUIT SAT *with $\Sigma_2\mathsf{P}$-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 $\mathsf{NP} \neq \mathsf{NC}^1$.

(Here, we use the LOGTIME-uniform definition of NC^1.) The choice of "Σ_2" in the theorem statement is somewhat arbitrary: using "Σ_c" for any $c \geq 2$ would suffice. To prove this theorem, we show that $\mathsf{NP} = \mathsf{NC}^1$ implies a strong circuit lower bound on Q-CIRCUIT SAT, even for circuits with an oracle in the polynomial hierarchy.

We also prove as a consequence of self-improvement (Theorem 5.2) 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 $\mathsf{NP} \neq \mathsf{NC}^1$. We should stress that we do not consider these theorems as a viable approach to separating NP from NC^1; rather, it indicates to us that proving the hypotheses seems overly strong and unlikely.

1.1 Open Problems

We pose here a few intriguing open problems, to ensure that the reader sees them.

- **Could self-improvement go all the way down to $\mathsf{P} = \mathsf{NP}$?** Is it possible that (say) linear-time SAT algorithms for exponential-size circuits might imply polynomial-time algorithms for polynomial-size circuits, concluding $\mathsf{P} = \mathsf{NP}$? 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 $\mathsf{NP} \subset \mathsf{P/poly}$?

- **Could self-improvement-style results hold for other combinatorial problems, besides just circuit-based ones?** It seems crucial for the self-improvement results that the algorithm solving the problem can be modeled extremely efficiently, within an instance of the problem. (Perhaps this alone shouldn't be considered an impediment, given the ubiquity of complete problems for many complexity classes.)

- **Can the uniform $\mathsf{SYM} \circ \mathsf{SYM}$ circuit lower bounds be further improved?** In principle, some fast matrix multiplication algorithms can be implemented in TC^0 [PPJA18] 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.

2 Preliminaries

We assume familiarity with computational complexity, especially circuit complexity [Vol99, AB09, Juk12]. 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 [Vol99] 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 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 [Val76], Pippenger-Fischer [PF79])**.** 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 [SW22], 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 1.1) 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 3.5, we obtain a subexponential-time algorithm for Σ_kCIRCUIT SAT, for every constant k.)

Other works have demonstrated phenomena which are similar to our self-improvement results, but differ in various critical ways. Williams [Wil13] 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 Π 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 poly$(\log n)^d$ space for all $c, d \geq 1$, then a dramatic speed-up is possible: every such Π can be solved nondeterministically in $O(n^3)$ time, which would imply $\mathsf{LOGSPACE} \neq \mathsf{NP}$ 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. [Wil13] 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.[7] 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 [PP10] study OPP algorithms, which are probabilistic polynomial time algorithms with $1/p(n)$ success probability, where $p(n)$ can be exponential in n. They show that 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 CIRCUIT SAT algorithms (Theorem B.1).

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 [Sri03, AK10, LW13, OS18, MMW19, OPS19, CMMW19, CJW19, CHO+20], 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 [AK10] show that if Boolean Formula Evaluation has constant-depth MAJORITY/NOT (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, we only really need 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 [KF77]; further related references include [BG93, BBG98, CC97, FGW06, Wil13].

[7] 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 $\varepsilon > 0$.

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 α, β 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 [Wil14, AWY15, CW21]. 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 $2^{\beta m+o(m)} \leq 2^{\beta n/(1+\alpha)+o(n)}$ time, 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 δn 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 δ 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

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

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 kth 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 $f_k \geq 0$ 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 $k(n) \geq \omega(1)$, we have $f_{k(n)} \leq o(1)$. For the case where $\alpha < \beta$, we have $f_0 - f_1 = \beta\alpha/(1+\alpha)$, and

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

where the last inequality follows since $f_k + \alpha$ and $f_{k-1} + \alpha$ are both at least β (the sequence $\{f_k\}$ is monotone non-decreasing). Therefore for $\beta > \alpha$ and any function $k(n) \geq \omega(1)$, $f_{k(n)}$ 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 (Appendix A)**.** *Suppose there are $\beta \geq \alpha > 0$ 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 $\alpha \leq \beta$. 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 $\alpha \leq \beta$. 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 1.4 shows one major consequence that would follow if we could strengthen self-improvement so that the algorithm uses 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 1.2. *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 [DKKY10]). 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

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

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

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:

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

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. One reason why Theorem 1.1 should be considered surprising is that the reduction seems to be in the "wrong direction": intuitively, the Circuit SAT problem only gets *easier* as the circuit size *increases*. (The 2^n search space becomes "smaller" relative to a larger input.)

The ideas of Theorem 1.1 lead to an intriguing equivalence for nearly-linear time Large Circuit SAT algorithms. For simplicity, we state the result in terms of algorithms solving Circuit SAT, but it also applies to non-uniform and randomized algorithms (Appendix B), as well as #Circuit SAT and Q-Circuit SAT.

Let ε-Large Circuit SAT be the problem of checking satisfiability for circuits of size N with $\varepsilon \log(N)$ inputs.

Theorem 3.4. *The following are equivalent:*

(1) There is an $\varepsilon \in (0, 1)$ such that ε-Large Circuit SAT is in $N^{1+o(1)}$ time.

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

Theorem 3.4 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 3.4 can be seen as a strengthening of Corollary 1.1.

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 $\alpha \leq \varepsilon$. 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 $\alpha \leq t\varepsilon$. 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 satisifability that correspond to constant levels of the polynomial hierarchy. We naturally define Σ_k ε-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 "Σ_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 Σ_1 ε-LARGE CIRCUIT SAT is equivalent to ε-LARGE CIRCUIT SAT, and Σ_k α-LARGE CIRCUIT SAT corresponds to a polynomial-time solvable version of the Σ_kP-complete problem Σ_k-SAT [AB09].

Theorem 3.5 (Extension of Theorem 3.4)**.** *The following are equivalent:*

(1) There is an $\varepsilon \in (0,1)$ such that ε-LARGE CIRCUIT SAT is in $N^{1+o(1)}$ time.

(2) For **every** *$\alpha > 0$ (including arbitrarily large α), α-LARGE CIRCUIT SAT is in $N^{1+o(1)}$ time.*

(3) For every $k \geq 1$ and every $\alpha > 0$, Σ_k α-LARGE CIRCUIT SAT is in $N^{1+o(1)}$ time.

Proof. (1) $\iff$ (2) follows from Theorem 3.4. (3) implies (2) by setting $k = 1$.

We prove that (2) implies (3). Given an instance C of Σ_kCIRCUIT 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 ℓ 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 3.4, 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 Σ_kCIRCUIT 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 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 1.1. *ETH implies that, for* **every** $\varepsilon > 0$, ε-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 1.2. *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* $2^{\alpha n + (1-\varepsilon)n + o(n)}$.

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. □

3.2 Self-Improvement for Random-Access Models

So far, the self-improving results we have proved hold for any model of computation such that time-T algorithms can be simulated by circuits of $T^{1+o(1)}$-size, such as multitape Turing machines; the results also hold for solving CIRCUIT SAT with circuit families. However, for some random-access models (RAMs) of computation, it is only known (for example) that time-T algorithms can be simulated by $T^{2+o(1)}$-size circuits. In this section, we sketch how to prove self-improvement results in such models as well, in such a way that we can still refute ETH under analogous conditions. Fixing any random-access model of computation, and fixing any constant $\varepsilon > 0$, we define the following problem:

Problem: RAM SAT_ε
Given: A parameter N given in unary, along with a random-access machine M of description length at most $O(\log_2(N))^2$.
Decide: Is there an $x \in \{0,1\}^{\varepsilon \log_2(N)}$ such that $M(x)$ accepts in at most N steps?

Analogously to LARGE CIRCUIT SAT, where there is a circuit of size N on $\varepsilon \log_2(N)$ inputs, and we wish to find a satisfying assignment, we are given the description of a RAM M running up to N steps, and we wish to find a $(\varepsilon \log_2(N))$-bit input x that makes M accept. As before, the obvious algorithm for RAM SAT_ε runs in $\tilde{O}(N^{1+\varepsilon})$ time (omitting polylog factors that may depend on the particular random-access model). Applying precisely the same proof as in Theorem 1.1, we can obtain a self-improvement result for RAM SAT. For illustration, we just give a special case. Define the problem SUBEXP RAM SAT to be:

> Given n and a RAM M of description length at most $O(n^2)$ that runs in up to $2^{o(n)}$ steps on inputs of length n, decide if there is an input x of length n such that $M(x)$ accepts.

Clearly, the obvious algorithm for SUBEXP RAM SAT runs in $2^{n+o(n)}$ time.

Theorem 3.6. *Suppose there is an $\varepsilon > 0$ such that* RAM SAT_ε *can be solved on random-access machines in $N^{1+o(1)}$ time. Then for every $\varepsilon > 0$,* SUBEXP RAM SAT *can be solved in $2^{\varepsilon n}$ time.*

Proof. Assume there exists a RAM M' that can solve RAM SAT_ε in $N^{1+o(1)}$ time. We prove by induction that for all natural numbers k, SUBEXP RAM SAT is solvable in $2^{f_k n+o(n)}$ time, where

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

The base case is obvious. Let M_0 be the RAM (of description length $O(1)$) that runs the $2^{n+o(n)}$-time brute-force algorithm for SUBEXP RAM SAT on a given input $\langle n, M\rangle$, where the length of the description of M is at most n^2.

For the inductive step, suppose we are given an instance $\langle n, M\rangle$ of SUBEXP RAM SAT. Conceptually split the n-bit input x into $A = \varepsilon f_{k-1} n/(1+\varepsilon f_{k-1})$ bits and $B = n/(1+\varepsilon f_{k-1})$ bits, noting that $A+B = n$. Build a RAM M_k that, given an A-bit input y, constructs a RAM M_y that runs M with y hard-coded in its first A input bits, and calls the RAM M_{k-1} on $\langle B, M_y\rangle$, outputting its answer. Observe that the length of M_y, $|\langle M_y\rangle|$, is at most $|\langle M\rangle| + n + O(1)$, and that $|\langle M_k\rangle| \le |\langle M\rangle| + |\langle M_{k-1}\rangle| + O(1)$.

By induction, M_{k-1} takes $2^{f_{k-1}B+o(n)}$ time on $\langle B, M_y\rangle$. Let $N_k := 2^{f_{k-1}B}$. Since $A = \varepsilon f_{k-1} B$, we have $A = \varepsilon \log_2(N_k)$, so the RAM M_k runs in $N_k^{1+o(1)}$ time on $\varepsilon \log_2(N_k)$-bit input. By assumption, feeding $\langle M_k\rangle$ to the RAM M', its answer will determine SUBEXP RAM SAT for $\langle n, M\rangle$ in time

$$2^{f_{k-1}B+o(n)} = 2^{f_{k-1}n/(1+\varepsilon f_{k-1})+o(n)} = 2^{f_k n+o(n)}.$$

For every $k \ge 1$, we have

$$f_k = 1/(1+\varepsilon \cdot k),$$

as by induction we can infer

$$f_k = \frac{f_{k-1}}{1+\varepsilon f_{k-1}} = \frac{1/(1+\varepsilon\cdot(k-1))}{1+\varepsilon/(1+\varepsilon\cdot(k-1))} = \frac{1}{1+\varepsilon\cdot(k-1)+\varepsilon} = 1/(1+\varepsilon\cdot k).$$

Therefore, by setting $k := c/\varepsilon$ for arbitrarily large $c \ge 1$, we can solve SUBEXP RAM SAT in time $2^{n/(1+c)+o(n)}$. □

The astute reader may wonder why we let the machine description have length quadratic in its input length. The reason is that this constraint allows us to refute ETH with a faster algorithm for SUBEXP RAM SAT. (In fact, allowing a description of length $\Omega(n \log n)$ would suffice.)

Theorem 3.7. *Suppose there is an $\varepsilon > 0$ such that* RAM SAT_ε *can be solved on random-access machines in $N^{1+o(1)}$ time. Then ETH is false.*

Proof. We show how the hypothesis allows us to solve 3-SAT in $2^{\varepsilon n}$ time for every ε. Let $\varepsilon > 0$. Given a 3-CNF formula F on n variables and $O(n^3)$ clauses, we first apply the Sparsification Lemma [IPZ01] to reduce F to an OR of $t \leq 2^{\varepsilon n/2}$ 3-CNFs $F_1, \ldots, F_t$, where each F_i has at most $c_\varepsilon n$ clauses for a constant c_ε depending only on ε.

For all $i = 1, \ldots, t$, define a RAM M_i which takes n bits of input x, and accepts if and only if the formula F_i is satisfied by the n-variable assignment encoded by x. Since each F_i has at most $c_\varepsilon n$ clauses of width 3, each F_i can be encoded in at most $O(n \log n)$ bits. Therefore each M_i can be defined so that its description length is at most $O(n^2)$. By Theorem 3.6, SUBEXP RAM SAT on each M_i can be solved in time at most $O(2^{\varepsilon n/2})$, thereby deciding satisfiability for the formula F_i. As there are $t \leq 2^{\varepsilon n/2}$ such M_i's, we can determine satisfiability of the original F in time at most $O(2^{\varepsilon n})$ by forming each $M_1, \ldots, M_t$ in turn and running our assumed algorithm for SUBEXP RAM SAT on each M_i. □

4 Uniform Lower Bounds For Linear Time

In this section, we establish new unconditional lower bounds for linear time computation.

Reminder of Theorem 1.3. *There are linear-time decision problems which do not have POLYLOGTIME-uniform* $\mathsf{SYM} \circ \mathsf{SYM}$ *circuits of n^c gates, for all $c < 1.199$.*

Proof. Assume the opposite. Our goal will be to contradict the time hierarchy theorem ([AB09]). In particular, we will contradict the fact that there is a decision problem $f : \{0,1\}^\star \to \{0,1\}$ such that:

- for all n, and for all $x \in \{0,1\}^n \setminus \{0^n\}$, $f(x) = 0$.
- f is computable in 2^n time.
- f is not computable in $2^n/n^{10}$ time.

In terms of languages and complexity classes, we are saying that there is a unary language in $\mathsf{TIME}[2^n]$ that is not in $\mathsf{TIME}[2^n/n^{10}]$.[8] Let M be an algorithm computing f in time 2^n. Our fast simulation of M first checks if the input is of the form 0^n; if not, the simulation immediately rejects in linear time. It remains to demonstrate how to simulate M on 0^n.

Assuming that every linear-time problem has uniform $\mathsf{SYM} \circ \mathsf{SYM}$ circuits of size n^c, it follows from a standard padding argument that the 2^n-time machine M can be simulated by an $\mathsf{SYM} \circ \mathsf{SYM}$ circuit family $\{C_n\}$ where C_n has at most 2^{cn} gates. The family $\{C_n\}$ is also POLYLOGTIME-uniform; let us describe in detail what this means (although the specific uniformity conditions given here are not critical for the argument).

[8] The argument could still be carried out without assuming the hard language is unary, but the use of a unary language simplifies some parts of the proof.

- First, there is a constant $a \geq 1$ and an n^a-time algorithm A which on an input $\langle x, n \rangle$ of $cn + O(\log n)$ bits, outputs gate information for the x-th gate on the bottom layer of the circuit C_n, where x is interpreted as an integer ranging from 1 to 2^{cn}. For concreteness, let us assume that A outputs an n-bit vector indicating which of the n inputs to C_n have wires into the x-th gate, and A outputs a table $T_x[0], \ldots, T_x[n]$ of $n+1$ bits that describes the symmetric function computed at the x-th gate. (The interpretation is that when $i \in \{0, \ldots, n\}$ inputs to the x-th gate are true, the symmetric function outputs $T_x[i]$.)

- To handle the top SYM gate, we also have an n^a-time algorithm B which, given an integer in $\{0, \ldots, 2^{cn}\}$ encoded in $cn + O(1)$ bits, outputs 0 or 1. This B computes the top SYM gate of the circuit, given the number of its inputs which are 1. (We assume that every SYM gate on the bottom layer has a wire to the top SYM gate.)

Using the algorithm A, in $\text{poly}(n^a)$ time we can construct a circuit D_n of $\text{poly}(n^a)$ size which takes a $cn + O(1)$ bit string x, and outputs 1 if and only if the x-th gate in the bottom SYM layer outputs 1 on the input 0^n. This can be done by simply calling A on $\langle x, n \rangle$ and outputting the value $T_x[0]$, since none of the n bits of input are set true.

Observe that computing #CIRCUIT SAT on the circuit D_n is equivalent to determining the number of gates on the bottom SYM layer which output 1, which equals the number of ones going into the top SYM gate. Feeding this number into the algorithm B outputs the value of the SYM $\circ$ SYM circuit on the input 0^n. Therefore, if we can compute #CIRCUIT SAT on D_n in less than $2^n/n^{10}$ time, the time hierarchy theorem used above will be contradicted.

We will show how to solve #CIRCUIT SAT on D_n more efficiently, applying (again) the assumption that linear time has uniform SYM $\circ$ SYM circuits, and using a key idea from Theorem 1.2. Let $\gamma \in (0, 1)$ be a parameter to be set later. Given the circuit D_n, in $2^{\gamma n + o(n)}$ time we can construct a circuit E which takes in a $cn - \gamma n + O(1)$ bit input z, and computes the integer sum

$$\sum_{y \in \{0,1\}^{\gamma n}} D_n(yz).$$

This circuit E has $t \leq O(n)$ output bits encoding the sum; we form circuits $E_{t-1}, \ldots, E_0$ for each of the t output bits of the sum, where E_0 outputs the low-order bit, E_{t-1} outputs the high-order bit, and so on. By our assumption that linear-time has uniform SYM $\circ$ SYM circuits of size n^c, for each circuit E_i, we can construct an equivalent SYM $\circ$ SYM circuit F_i, of $2^{c\gamma n + o(n)}$ size with $cn - \gamma n + O(1)$ inputs. Furthermore, each F_i can be constructed in $2^{c\gamma n + o(n)}$ time, by calling the relevant uniform algorithms which produce gate information.

If we compute #CIRCUIT SAT for each F_i, we can determine #CIRCUIT SAT for the original circuit D_n by computing

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

analogously as in Theorem 1.2. We now turn to computing #CIRCUIT SAT for a given F_i.

Let $s = 2^{c\gamma n + o(n)}$. Following a similar construction in [Wil18], at this point we can do something stronger than solve #CIRCUIT SAT: we can evaluate each s-size F_i on all $2^{cn - \gamma n}$ possible assignments, by using fast matrix multiplication. WLOG, let $n' := cn - \gamma n$ be even, and split the set of n' input variables into two parts P_1 and P_2 of $n'/2$ variables each. Enumerate all $2^{n'/2}$ possible assignments to P_1 and P_2,

creating lists L_1 and L_2 of $2^{n'/2}$ length each. From these lists, we form Boolean matrices M_1 and M_2, as follows. Index the rows of M_1 by assignments in L_1, and the columns by pairs in $[n] \times [s]$, defining

$$M_1[a, (i,j)] = 1 \Leftrightarrow \text{partial assignment } a \text{ sets exactly } i \text{ inputs true going into the } j\text{-th (bottom) SYM gate.}$$

Index the rows of M_2 by $[n] \times [s]$ and the columns by assignments in L_2, defining

$$M_2[(i,j), a'] = 1 \Leftrightarrow \text{ the } j\text{-th (bottom) SYM gate outputs 1 on partial assignment } a', \text{ assuming that exactly } i \text{ other inputs from the other partial assignment are true.}$$

Each entry of M_1 and M_2 can be computed in $\text{poly}(n)$ time. Observe that the full variable assignment $(a, a') \in L_1 \times L_2$ makes the j-th SYM gate output 1, if and only if there is an i such that $M_1[a, (i,j)] \wedge M_2[(i,j), a'] = 1$. Furthermore, such an i is always unique when it exists (a partial assignment a cannot set exactly i and exactly i' inputs true, where $i \neq i'$). Therefore the (a, a') entry of the matrix product $M_1 \cdot M_2$ equals the number of (bottom) SYM gates which are true on the assignment (a, a'). By feeding each entry of the matrix product $M_1 \cdot M_2$ into the top SYM gate of F_i, we can evaluate F_i on all possible inputs, and thereby determine #CIRCUIT SAT for F_i. This completes the description of the simulation of M, modulo the setting of the parameter $\gamma > 0$.

Now, we wish to determine the conditions under which all of the above can be implemented in $2^n/n^{10}$ time, or less. The bottleneck is the matrix multiplication step: the dimensions of the matrices are

$$2^{(cn-\gamma n)/2} \times 2^{c\gamma n + o(n)} \text{ and } 2^{c\gamma n + o(n)} \times 2^{(cn-\gamma n)/2}.$$

(Note that c appears in the outer dimension of the matrices due to the first application of the assumption, and c appears in the inner dimension due to the second application.)

Our goal is to now maximize c (setting $\gamma > 0$ appropriately) such that the overall matrix multiplication takes less than $2^n/n^{10}$ time. As a warm-up, assuming $N \times N$ matrix multiplication can be achieved in $N^{2+o(1)}$ time, we can set $(c-\gamma)/2 = c\gamma$. This has the solution $\gamma = c/(2c+1)$, so the matrix multiplication will take $2^{2c^2 n/(2c+1)} + o(n)$ time. In that case, we need $c^2 < (2c+1)/2$ in order to obtain a contradiction, which happens precisely when $c < (1+\sqrt{3})/2 < 1.36$.

In general, suppose we have an algorithm for multiplying matrices of dimensions $N \times N^\alpha$ and $N^\alpha \times N$ that runs in $N^{\beta + o(1)}$ time. Setting $\gamma > 0$ so that

$$c\gamma = \alpha \cdot (c - \gamma)/2,$$

i.e., $\gamma = \alpha c/(\alpha + 2c)$, the running time of applying the assumed rectangular matrix multiplication algorithm will be

$$2^{\beta(c-\gamma)n/2 + o(n)} = 2^{\beta(c - \alpha c/(\alpha+2c))n/2}.$$

To get a contradiction for the largest possible c, we want to set α, β that maximize c subject to the constraint

$$\beta\left(c - \frac{\alpha c}{\alpha + 2c}\right) < 2.$$

Solving the resulting quadratic in terms of c, the above constraint is equivalent to

$$c < \frac{1 + \sqrt{\alpha\beta + 1}}{\beta}.$$

Examining the table of best-known α, β values (Table 3) in [LU18], we find that the RHS of the inequality is maximized for $\alpha = 0.8$ and $\beta = 2.222256$, in which case $c < 1.19998$ suffices. □

5 Lower Bounds From Improving Self-Improvement

We noted in Section 3 that our self-improvement results are relativizing, if oracles are placed in both the problem definition and in the algorithms solving the problem. Here, we will show that if self-improvement could be relativized in a different way, then we would separate NP from NC^1. (As mentioned in the introduction, we do not consider this to be an approach to separating NP from NC^1, but rather as a potential limitation to pushing the techniques further.)

First, we prove that if Q-CIRCUIT SAT has subexponential-size circuits, then $\mathsf{NP} \neq \mathsf{NC}^1$.

Theorem 5.1. *Suppose there is a $\ell \geq 0$ such that for all $\varepsilon > 0$,* Q-CIRCUIT SAT *on $2^{o(n)}$-size circuits has $2^{\varepsilon n + o(n)}$-size non-uniform $\Sigma_\ell SAT$-oracle circuits. Then* $\mathsf{NP} \neq \mathsf{NC}^1$.

(We define $\Sigma_0 SAT$ to simply be the CIRCUIT EVAL decision problem.)

Proof. For a contradiction, assume Q-CIRCUIT SAT on $2^{o(n)}$-size circuits has $2^{\varepsilon n + o(n)}$-size (non-uniform) $\Sigma_c\mathsf{P}$-oracle circuits, and assume $\mathsf{NP} = \mathsf{NC}^1$.

If $\mathsf{NP} = \mathsf{NC}^1$, then the entire polynomial hierarchy can be simulated in NC^1 by standard arguments ([AB09]), so $\Sigma_\ell\mathsf{P} = \mathsf{NC}^1$ for all $\ell \geq 1$. By a standard padding/translation argument, we have $\Sigma_\ell\mathsf{E} = \mathsf{TIME}[2^{O(n)}] = \mathsf{ATIME}[O(n)]$, where the latter class denotes *alternating* linear time. However, it is well-known that functions $f' : \{0,1\}^\star \to \{0,1\}$ of maximum circuit complexity $\Theta(2^n/n)$ reside in $\Sigma_3\mathsf{E}$ ([AB09, Juk12]). Therefore there is a constant $k' \geq 1$ and a function $f' : \{0,1\}^\star \to \{0,1\}$ that requires $\Omega(2^n/n)$-size circuits, but f' is computable in alternating time $k' \cdot n$. Furthermore, the construction of f' relativizes, so there is also a function $f : \{0,1\}^\star \to \{0,1\}$ in $\Sigma_{\ell+3}\mathsf{E}$ with maximum $\Sigma_\ell SAT$-oracle circuit complexity, and since $\Sigma_{\ell+3}\mathsf{E} = \mathsf{ATIME}[O(n)]$, this f is computable in alternating time kn for some constant k. Let M be the alternating machine computing f.

We observe that standard methods for reducing computations into propositional logic ([AB09]) show that the quantified Boolean formula problem (TQBF) is strongly complete for alternating linear time, in the following sense: for every alternating machine M running in kn time, there is a constant $c \geq 1$ (depending only on the tape alphabet of M) such that the computation of M on an input x of length n can be reduced to a QBF instance $\Phi_{M,x}$ having at most ckn quantified variables, with a circuit predicate of $\mathrm{poly}(n)$ size; furthermore, the construction of $\Phi_{M,x}$ takes only $\mathrm{poly}(n)$ time and its encoding has $\mathrm{poly}(n)$ bits. Moreover, without loss of generality, the reduction can be designed so that the n bits of the input x always appear in n specific locations of the encoding of the QBF $\Phi_{M,x}$.

Finally, we use the presumed $2^{\varepsilon n + o(n)}$-size (non-uniform) $\Sigma_\ell SAT$-oracle circuits for Q-CIRCUIT SAT on n-variable instances of size $2^{o(n)}$. Given such a circuit C_n that will take a Q-CIRCUIT SAT instance of size $2^{o(n)}$, we feed C_n a description of $\Phi_{M,x}$ with kcn variables, *but we leave the n bits describing x unassigned, as free variables*. That is, the resulting circuit C'_n, which is C_n having all but n bits set by the description of $\Phi_{M,x}$, is a circuit with n inputs, such that $C'_n(x)$ outputs the truth value of $\Phi_{M,x}$.

By construction, C'_n computes f. However, C'_n has size $2^{\varepsilon kcn + o(n)}$. Since k, c are fixed constants, and $\varepsilon > 0$ can be made arbitrarily small, this contradicts the fact that f does not have $\Sigma_\ell SAT$-oracle circuits of $2^{\alpha n}$ size for all $\alpha < 1$. This concludes the proof. □

Let us give two relevant corollaries of Theorem 5.1 for self-improvement. The first states that non-uniform algorithms that can efficiently decide Q-CIRCUIT SAT on *any* large circuit sizes would separate NP from NC^1.

Theorem 5.2. *Suppose there is some $k > 0$ such that for all $\varepsilon > 0$,* Q-CIRCUIT SAT *on 2^{kn}-size circuits has $2^{kn+\varepsilon n}$-size non-uniform circuits. Then* $\mathsf{NP} \neq \mathsf{NC}^1$.

Proof. Given the hypothesis of the theorem, the self-improvement theorem for Q-CIRCUIT SAT (Theorem 1.2) implies that Q-CIRCUIT SAT on $2^{o(n)}$-size circuits has $2^{\varepsilon n+o(n)}$-size non-uniform circuits. The consequence follows from Theorem 5.1. □

The second corollary shows that "improving self-improvement" to hold for oracles in the polynomial hierarchy (without changing the underlying inputs) would also separate NP from NC^1.

Reminder of Theorem 1.4. *Suppose self-improvement holds for* Q-CIRCUIT SAT *with* $\Sigma_2\mathsf{P}$*-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 there is a* $c \geq 0$ *such that for all* $\varepsilon > 0$*,* Q-CIRCUIT SAT *on* $2^{o(n)}$*-size circuits has* $2^{\varepsilon n+o(n)}$*-size non-uniform* $\Sigma_c SAT$*-oracle circuits.*

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

Proof. By Proposition 1, we already know that Q-CIRCUIT SAT on $2^{n+o(n)}$-size circuits in $2^{n+o(n)}$ time with a $\Sigma_2 SAT$ oracle. Thus by assumption, there is a $c > 0$ such that for all $\varepsilon > 0$, Q-CIRCUIT SAT on $2^{o(n)}$-size circuits has $2^{\varepsilon n+o(n)}$-size non-uniform $\Sigma_c SAT$-oracle circuits. Applying Theorem 5.1, $\mathsf{NP} \neq \mathsf{NC}^1$ follows. □

Although we do not believe the above is currently a viable approach to separating NP from NC^1, we do think the above results should give a minor pause. It is not obvious whether we should believe that non-uniform $\Sigma_c SAT$-oracle circuits exist for Q-CIRCUIT SAT. It seems highly unlikely that there are subexponential-time *algorithms* for TQBF. However, once we are allowed non-uniformity and oracles in the polynomial hierarchy, the situation becomes less clear.

Acknowledgements. I am grateful to Lijie Chen, Russell Impagliazzo, Valentine Kabanets, Mohan Paturi, Rahul Santhanam, and Michael Wehar for interesting discussions. I also thank Shyan Akmal for comments and corrections on an earlier version of the manuscript. Many thanks to the Simons Institute for providing a stimulating environment during the Meta-Complexity and Satisfiability Extended Reunion programs.

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 [ACF+22]). Namely, Prof. Hartmanis often asked me: "If $\mathsf{P} = \mathsf{NP}$, 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

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[ACF+22] Eric Allender, Jin-Yi Cai, Lance Fortnow, William Gasarch, Neil Immerman, Stuart Kurtz, James Royer, and Ryan Williams. Open problems column: Open problems by or inspired by Juris Hartmanis. *SIGACT News*, 53(4):26, 2022.

[ACW16] Josh Alman, Timothy M. Chan, and R. Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In *FOCS*, pages 467–476, 2016.

[AG94] Eric Allender and Vivek Gore. A uniform circuit lower bound for the permanent. *SIAM J. Comput.*, 23(5):1026–1049, 1994.

[AHWW16] Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with edit distance and friends: or: a polylog shaved is a lower bound made. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 375–388. ACM, 2016.

[AK10] Eric Allender and Michal Koucký. Amplifying lower bounds by means of self-reducibility. *J. ACM*, 57(3):14:1–14:36, 2010.

[AKR+01] Eric Allender, Michal Koucký, Detlef Ronneburger, Sambuddha Roy, and V. Vinay. Time-space tradeoffs in the counting hierarchy. In *CCC*, pages 295–302. IEEE Computer Society, 2001.

[AWY15] Amir Abboud, Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *SODA*, pages 218–230. Society for Industrial and Applied Mathematics, 2015.

[BBG98] Stephen A. Bloch, Jonathan F. Buss, and Judy Goldsmith. Sharply bounded alternation and quasilinear time. *Theory Comput. Syst.*, 31(2):187–214, 1998.

[BG93] Jonathan F. Buss and Judy Goldsmith. Nondeterminism within P. *SIAM J. Comput.*, 22(3):560–572, 1993.

[BGG+22] Vishwas Bhargava, Sumanta Ghosh, Zeyu Guo, Mrinal Kumar, and Chris Umans. Fast multivariate multipoint evaluation over all finite fields. In *FOCS*, pages 221–232. IEEE, 2022.

[BGKM22] Vishwas Bhargava, Sumanta Ghosh, Mrinal Kumar, and Chandra Kanta Mohapatra. Fast, algebraic multivariate multipoint evaluation in small characteristic and applications. In *STOC*, pages 403–415. ACM, 2022.

[BKW15] Mark Braverman, Young Kun-Ko, and Omri Weinstein. Approximating the best nash equilibrium in $n^{o(\log n)}$-time breaks the exponential time hypothesis. In *SODA*, pages 970–982, 2015.

[BM74] Allan Borodin and R. Moenck. Fast modular transforms. *J. Comput. Syst. Sci.*, 8(3):366–386, 1974.

[BW15] Samuel R. Buss and Ryan Williams. Limits on alternation trading proofs for time-space lower bounds. *Comput. Complex.*, 24(3):533–600, 2015.

[CC97] Liming Cai and Jianer Chen. On the amount of nondeterminism and the power of verifying. *SIAM J. Comput.*, 26(3):733–750, 1997.

[CHO+20] Lijie Chen, Shuichi Hirahara, Igor Carboni Oliveira, Ján Pich, Ninad Rajgopal, and Rahul Santhanam. Beyond natural proofs: Hardness magnification and locality. In *ITCS*, pages 70:1–70:48, 2020.

[CIP06] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. A duality between clause width and clause density for SAT. In *CCC*, pages 252–260, 2006.

[CJW19] Lijie Chen, Ce Jin, and R. Ryan Williams. Hardness magnification for all sparse NP languages. In *FOCS*, pages 1240–1255, 2019.

[CMMW19] Lijie Chen, Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. Relations and equivalences between circuit lower bounds and Karp-Lipton theorems. In *CCC*, pages 30:1–30:21, 2019.

[CPP16] Marek Cygan, Marcin Pilipczuk, and Michal Pilipczuk. Known algorithms for edge clique cover are probably optimal. *SIAM J. Comput.*, 45(1):67–83, 2016.

[CRTY20] Lijie Chen, Ron D. Rothblum, Roei Tell, and Eylon Yogev. On exponential-time hypotheses, derandomization, and circuit lower bounds: Extended abstract. In *FOCS*, pages 13–23. IEEE, 2020.

[CT65] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[CW21] Timothy M. Chan and R. Ryan Williams. Deterministic apsp, orthogonal vectors, and more: Quickly derandomizing Razborov-Smolensky. *ACM Trans. Algorithms*, 17(1):2:1–2:14, 2021.

[DHM+14] Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslaman, and Martin Wahlen. Exponential time complexity of the permanent and the tutte polynomial. *ACM Trans. Algorithms*, 10(4):21:1–21:32, 2014.

[DKKY10] Evgeny Demenkov, Arist Kojevnikov, Alexander S. Kulikov, and Grigory Yaroslavtsev. New upper bounds on the boolean circuit complexity of symmetric functions. *Inf. Process. Lett.*, 110(7):264–267, 2010.

[FGW06] Jörg Flum, Martin Grohe, and Mark Weyer. Bounded fixed-parameter tractability and $\log^2 n$ nondeterministic bits. *J. Comput. Syst. Sci.*, 72(1):34–71, 2006.

[Fid72] Charles M. Fiduccia. Polynomial evaluation via the division algorithm: The fast Fourier transform revisited. In *STOC*, pages 88–93, 1972.

[FLvMV05] Lance Fortnow, Richard J. Lipton, Dieter van Melkebeek, and Anastasios Viglas. Time-space lower bounds for satisfiability. *J. ACM*, 52(6):835–865, 2005.

[GGHN08] Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Data reduction and exact algorithms for clique cover. *ACM J. Exp. Algorithmics*, 13, 2008.

[GHH+23] Sumanta Ghosh, Prahladh Harsha, Simao Herdade, Mrinal Kumar, and Ramprasad Saptharishi. Fast numerical multivariate multipoint evaluation. *Electronic Colloquium on Computational Complexity*, TR23-033, 2023.

[IP99] Russell Impagliazzo and Ramamohan Paturi. Complexity of k-sat. In *Proceedings of the 14th Annual IEEE Conference on Computational Complexity, Atlanta, Georgia, USA, May 4-6, 1999*, pages 237–240. IEEE Computer Society, 1999.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

[Juk12] Stasys Jukna. *Boolean function complexity: advances and frontiers*, volume 27. Springer Science & Business Media, 2012.

[Kan83] Ravi Kannan. Alternation and the power of nondeterminism. In *STOC*, pages 344–346. ACM, 1983.

[KF77] Chandra M. R. Kintala and Patrick C. Fischer. Computations with a restricted number of nondeterministic steps (extended abstract). In *STOC*, pages 178–185. ACM, 1977.

[Ko82] Ker-I Ko. Some observations on the probabilistic algorithms and np-hard problems. *Inf. Process. Lett.*, 14(1):39–43, 1982.

[KU11] Kiran S. Kedlaya and Christopher Umans. Fast polynomial factorization and modular composition. *SIAM J. Comput.*, 40(6):1767–1802, 2011.

[KW16] Daniel M. Kane and Ryan Williams. Super-linear gate and super-quadratic wire lower bounds for depth-two and depth-three threshold circuits. In *STOC*, pages 633–643, 2016.

[LMM03] Richard J. Lipton, Evangelos Markakis, and Aranyak Mehta. Playing large games using simple strategies. In *Proceedings 4th ACM Conference on Electronic Commerce (EC)*, pages 36–41. ACM, 2003.

[LU18] Francois Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the coppersmith-winograd tensor. In *SODA*, pages 1029–1046. SIAM, 2018.

[LW13] Richard J. Lipton and Ryan Williams. Amplifying circuit lower bounds against polynomial time, with applications. *Computational Complexity*, 22(2):311–343, 2013.

[MMW19] Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. Weak lower bounds on resource-bounded compression imply strong separations of complexity classes. In *STOC*, pages 1215–1225. ACM, 2019.

[MW21] Abhijit Mudigonda and R. Ryan Williams. Time-space lower bounds for simulating proof systems with quantum and randomized verifiers. In *ITCS*, volume 185 of *LIPIcs*, pages 50:1–50:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[OPS19] Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. In *34th Computational Complexity Conference, CCC 2019*, pages 27:1–27:29, 2019.

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

[PF79] Nicholas Pippenger and Michael J. Fischer. Relations among complexity measures. *J. ACM*, 26(2):361–381, 1979.

[PP10] Ramamohan Paturi and Pavel Pudlák. On the complexity of circuit satisfiability. In Leonard J. Schulman, editor, *STOC*, pages 241–250. ACM, 2010.

[PPJA18] Ojas Parekh, Cynthia A. Phillips, Conrad D. James, and James B. Aimone. Constant-depth and subcubic-size threshold circuits for matrix multiplication. In Christian Scheideler and Jeremy T. Fineman, editors, *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures, SPAA 2018, Vienna, Austria, July 16-18, 2018*, pages 67–76. ACM, 2018.

[Rub16] Aviad Rubinstein. Settling the complexity of computing approximate two-player nash equilibria. In *FOCS*, pages 258–265, 2016.

[San23] Rahul Santhanam. An algorithmic approach to uniform lower bounds. *Electronic Colloquium on Computational Complexity (ECCC)*, TR23-028:100, 2023.

[Sri03] Aravind Srinivasan. On the approximability of clique and related maximization problems. *J. Comput. Syst. Sci.*, 67(3):633–651, 2003.

[SW22] András Z. Salamon and Michael Wehar. Superlinear lower bounds based on ETH. In *STACS*, volume 219 of *LIPIcs*, pages 55:1–55:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[Tam16] 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)*, TR16-100, 2016.

[Val76] Leslie G. Valiant. Universal circuits (preliminary report). In *Proceedings of the 8th Annual ACM Symposium on Theory of Computing*, pages 196–203. ACM, 1976.

[Vas18] Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians (ICM)*, pages 3447–3487. World Scientific, 2018.

[vM06] Dieter van Melkebeek. A survey of lower bounds for satisfiability and related problems. *Found. Trends Theor. Comput. Sci.*, 2(3):197–303, 2006.

[vMR05] Dieter van Melkebeek and Ran Raz. A time lower bound for satisfiability. *Theor. Comput. Sci.*, 348(2-3):311–320, 2005.

[vMW12] Dieter van Melkebeek and Thomas Watson. Time-space efficient simulations of quantum computations. *Theory Comput.*, 8(1):1–51, 2012.

[Vol99] Heribert Vollmer. *Introduction to Circuit Complexity - A Uniform Approach*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 1999.

[Wil13] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM J. Comput.*, 42(3):1218–1244, 2013.

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

[Wil18] R. Ryan Williams. New algorithms and lower bounds for circuits with linear threshold gates. *Theory of Computing*, 14(1):1–25, 2018.

A Appendix: A Slightly Stronger Self-Improvement

Reminder of Theorem 3.1. *Suppose there are $\beta \geq \alpha > 0$ 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.*

Here we merely sketch the argument, as it closely follows the proof of Theorem 1.1, but also tracks the dependence on circuit size. Start with a CIRCUIT SAT instance C of $2^{\gamma n}$ size and n inputs, for γ to be set later. Let S_ε be an algorithm running in $2^{\beta n+\varepsilon n}$ time that takes as input a description of a $2^{\alpha n+o(n)}$-size circuit C', and determines satisfiability for C'.

For a parameter $\delta \in (0,1)$ to be determined, build a circuit D on $m = (1-\delta)n$ inputs with the following behavior. First, given an assignment x of m bits, D plugs the bits of x into the first m inputs of C, yielding the description of a circuit $C' := C(x,\cdot)$ with $m' = n - m = \delta n$ inputs, and $2^{\gamma n+o(n)} \leq 2^{\gamma(1-\delta)m+o(m)}$ size. Next, D evaluates C' on all possible $2^{m'}$ assignments to the m' inputs of C' and takes the OR over all $2^{m'}$ outputs. Thus, D has size $2^{\delta n+\gamma(1-\delta)m+o(m)} \leq 2^{\delta m/(1-\delta)+\gamma(1-\delta)m+o(m)}$, and has m inputs.

If we set δ so that $\alpha = \delta/(1-\delta)+\gamma(1-\delta)$, the size of D will be precisely $2^{\alpha m+o(m)}$, and we can apply the algorithm S_ε. Setting $\gamma > 0$ appropriately, we can set $\delta \in (0,1)$. (In particular, the above constraint is equivalent to $\gamma = (\alpha(1-\delta)-\delta)/(1-\delta)^2$. For every $\alpha > 0$, one can find a $\gamma > 0$ such that we can satisfy the equation with some $\delta \in (0,1)$. If $\alpha \geq 1$, then .)

We feed the description of D to the assumed algorithm S_ε which runs in time $2^{\beta m+\varepsilon m} \leq 2^{\beta(1-\delta)n+\varepsilon(1-\delta)n}$ size, and outputs a yes/no answer. Observe that C is satisfiable if and only if $S(D)$ outputs yes. Now we know that satisfiability of circuits of $2^{\gamma n}$ size can be determined by an algorithm running in $2^{\beta(1-\delta)n+\varepsilon n}$ time, for every $\varepsilon > 0$. Denote this algorithm by $\mathcal{F}_1$.

We can repeat the argument again, but instead of taking an OR over all possible assignments, we call our new SAT algorithm instead. Suppose inductively that CIRCUIT SAT on n-input $2^{\gamma_k n}$-size circuits can be determined by circuits of $2^{f_k n+o(n)}$ size.

In particular, let $\delta_k \in (0,1)$ be a parameter, and we make a circuit D on $m = (1-\delta_k)n$ inputs with the following behavior: Given an assignment x of m bits, D plugs x into the first $(1-\delta_k)n$ inputs of C, yielding a circuit C' with $\delta_k n$ inputs and $2^{\gamma_{k+1} n}$ size. Setting $\gamma_{k+1} = \delta_k \cdot \gamma_k$, we can apply the induction hypothesis. Next, D calls the algorithm $\mathcal{F}_k$ to determine satisfiability of C', taking time $2^{\delta_k f_k n+o(n)}$. Now we have a circuit D on $m = (1-\delta_k)n$ inputs of size $2^{\delta_k \cdot f_k n+o(n)}$ which is equi-satisfiable to our original circuit C.

Setting δ_k such that $\delta_k \cdot f_k n = \alpha m$, our circuit D will have m inputs and $2^{\alpha m+o(m)}$ size, so its satisfiability can be determined in $2^{\beta m+\varepsilon m}$ time (for all $\varepsilon > 0$) by hypothesis. Setting $\delta_k = \alpha/(f_k+\alpha)$ accomplishes this. We can therefore determine satisfiability of C in time $2^{\beta m+\varepsilon m}$, for all $\varepsilon > 0$, which also implies a bound of $2^{\beta(1-\alpha/(f_k+\alpha))n+\varepsilon n}$, for all $\varepsilon > 0$. Let this algorithm be $\mathcal{F}_k$.

Setting $f_0 := \beta$, $f_{k+1} := \beta(1-\alpha/(f_k+\alpha))$, we can construct a sequence of algorithms $\mathcal{F}_k$ for computing satisfiability of $2^{\gamma_k n}$-size circuits, where the kth circuit has size $2^{f_k n+o(n)}$. For every $\alpha > 0$, we claim that the sequence $\{f_k\}$ is monotone decreasing, and converges to $f_\infty = \beta - \alpha$.

We have $\gamma_k := \gamma \cdot \left(\prod_{i=1}^{k} \delta_i\right)$. The sequence $\{\gamma_k\}$ is decreasing with increasing k, but all γ_k are greater than 0. We conclude that 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.

B Appendix: Extension to Randomized Algorithms

Here, we observe that the self-improvement results of this paper also hold for *randomized* algorithms solving CIRCUIT SAT, #CIRCUIT SAT, and Q-CIRCUIT SAT. (As usual, we need our model of randomized algorithms to have the property that for every $T(n)$-time algorithm, there is an equivalent $T(n)^{1+o(1)}$-size circuit for inputs of length n that can be generated in $T^{1+o(1)}$ time. These resulting "randomized circuits" will have n standard input bits, along with $O(T)$ auxiliary inputs to encode the randomness.) We begin by proving the result for CIRCUIT SAT.

Theorem B.1. *If there is a randomized* CIRCUIT SAT *algorithm running in $2^{\beta n+o(n)}$ time on $2^{\alpha n+o(n)}$-size circuits, then for all $\varepsilon > 0$, there is a randomized* CIRCUIT SAT *algorithm running in $2^{(\beta-\alpha+\varepsilon)}$ time on $2^{o(n)}$-size circuits.*

Proof. (Sketch) We sketch the simple modifications necessary to prove the result. Suppose there is a randomized algorithm S that runs in $2^{\beta n+o(n)}$ time on $2^{\alpha n+o(n)}$-size circuits and solves CIRCUIT SAT.

First of all, we may assume without loss of generality that our randomized algorithms never make a mistake when they report "SAT" (that is, all CIRCUIT SAT algorithms involved have one-sided error). Using the self-reducibility of SAT (in particular, applying the standard proof that $\mathsf{NP} \subseteq \mathsf{BPP}$ implies $\mathsf{NP} = \mathsf{RP}$ [Ko82]) by making $O(n)$ calls to the randomized CIRCUIT SAT algorithm (where n is the number of inputs to the circuit), we can force the randomized algorithm to compute a valid satisfying assignment, whenever one exists. If the original algorithm runs in T time, the new algorithm runs in $O(nT)$ time. (For all of our results, this factor of n is negligible, as the circuit size is always much larger than the number of inputs.) Moreover, we may assume that the error probability of S can be made below $1/n$ (for example), by simply repeating S for $O(\log n)$ trials, and outputting any satisfying assignment found during the trials.

We observe that the base case of our induction in Theorem 1.1 works with no modification: taking an OR over all assignments to some fraction of the variables is a deterministic operation that doesn't require any randomness. However, the new SAT algorithm $\mathcal{F}_1$ for small circuits that we obtain in the base case will be randomized (with one-sided error) and we will need to handle this property in the induction step.

Suppose that inductively, we have a randomized algorithm $\mathcal{F}_k$ for CIRCUIT SAT which takes the description of a circuit as well as some random bit-string r of length ℓ, and outputs a satisfying assignment when one exists with probability at least $9/10$. (Furthermore, if the given circuit is unsatisfiable, then $\mathcal{F}_k$ always outputs 0.) In the inductive step, when we construct a circuit D that has $(1-\delta)n$ inputs and calls $\mathcal{F}_k$ on the remaining simplified subcircuit with δn inputs, our circuit D now chooses $t = O(1)$ uniform independent random strings $r_1, \ldots, r_t$ of length ℓ, calls $\mathcal{F}_k$ on each of the strings r_i set as its randomness, and outputs any SAT assignment that is output by $\mathcal{F}_k$ over the t calls. (The circuit D can now be viewed as a *distribution* of circuits.) We have the following properties.

- If the original circuit C is satisfiable, then the (randomized) circuit D constructed is unsatisfiable with probability at most $1/10^t$: each of the (independent) t calls to $\mathcal{F}_k$ has probability at most $1/10$ of failing to output a SAT assignment, so the probability that all of them fail is at most $1/10^t$.
- If C is unsatisfiable, then D is unsatisfiable: all settings to the random strings of $\mathcal{F}_k$ and all settings to the input variables of D will result in a zero output, since $\mathcal{F}_k$ has one-sided error.

When we call the randomized algorithm S on the (randomized) circuit D, the algorithm S itself has an error probability of at most $1/n$. The probability of error overall for our new algorithm $\mathcal{F}_{k+1}$ is therefore at most $1/n + 1/10^t$, which can be made below $1/10$ by setting $t = 2$ for sufficiently large n. □

In fact, the above proof outline can be applied to self-improvement for randomized algorithms solving Q-CIRCUIT SAT and #CIRCUIT SAT as well (extending Theorem 1.2), but we need to be a little more careful. In particular, we do not know how to enforce one-sided error in a randomized algorithm for Q-CIRCUIT SAT, nor do we know how to do it for randomized algorithms for #CIRCUIT SAT.

However, suppose instead of making $t = O(1)$ calls to $\mathcal{F}_k$ as in the proof of Theorem B.1, the circuit D takes the majority vote of $t := cn$ calls to algorithm $\mathcal{F}_k$ with t independent random settings to the randomness of $\mathcal{F}_k$, where n is the number of input variables to the original circuit C, and $c \geq 1$ is a sufficiently large constant. The factor-of-n increase in circuit size resulting from making cn calls to $\mathcal{F}_k$ is still negligible, compared to the overall circuit sizes (which are mildly exponential: $2^{\delta n}$ size for some $\delta > 0$).

Fix a variable assignment a to the inputs of D. By a standard Chernoff-Hoeffding style tail bound, since each call to $\mathcal{F}_k$ has probability only $1/10$ of error, the probability that D returns an incorrect answer on a is the probability that our majority vote (of cn calls to $\mathcal{F}_k$) outputs an incorrect answer, which is at most $1/\exp(\Theta(cn))$. By a union bound over all possible $2^{(1-\delta)n}$ assignments to the $(1-\delta)n$ inputs of the circuit D, the probability that there is *some* variable assignment a to D such that D returns an incorrect output on a is still at most $1/\exp(n)$, by making the constant $c \geq 1$ large enough. Therefore, our reduction from the original circuit C to the randomized circuit D has probability of error only $1/\exp(n)$, and the rest of the analysis works as before.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il