Amplifying Lower Bounds by Means of Self-Reducibility[*]

Eric Allender[†]
Department of Computer Science
Rutgers University
New Brunswick, NJ 08855, USA
allender@cs.rutgers.edu

Michal Koucký[‡]
Institute of Mathematics of the
Academy of Sciences of the Czech Republic
Prague, Czech Republic
koucky@math.cas.cz

August 26, 2008

Abstract

We observe that many important computational problems in NC^1 share a simple self-reducibility property. We then show that, for any problem A having this self-reducibility property, A has polynomial size TC^0 circuits if and only if it has TC^0 circuits of size $n^{1+\epsilon}$ for every $\epsilon > 0$ (counting the number of wires in a circuit as the size of the circuit). As an example of what this observation yields, consider the Boolean Formula Evaluation problem (BFE), which is complete for NC^1 and has the self-reducibility property. It follows from a lower bound of Impagliazzo, Paturi, and Saks, that BFE requires depth d TC^0 circuits of size $n^{1+\epsilon_d}$. If one were able to improve this lower bound to show that there is some constant $\epsilon > 0$ such that every TC^0 circuit family recognizing BFE has size $n^{1+\epsilon}$, then it would follow that TC$^0 \neq$ NC^1. We show that proving lower bounds of the form $n^{1+\epsilon}$ is not ruled out by the Natural Proof framework of Razborov and Rudich and hence there is currently no known barrier for separating classes such as ACC^0, TC^0 and NC^1 via existing "natural" approaches to proving circuit lower bounds.

We also show that problems with small uniform constant-depth circuits have algorithms that simultaneously have small space and time bounds. We then make use of known time-space tradeoff lower bounds to show that SAT requires uniform depth d TC^0 and AC$^0[6]$ circuits of size n^{1+c} for some constant c depending on d.

1 Introduction

There is a great deal of pessimism in the research community, regarding the likelihood of proving superpolynomial lower bounds on the circuit size required for various computational problems. One goal of this paper is to suggest that there might be some reason to be more optimistic about prospects for circuit size lower bounds; we show that superpolynomial bounds would follow as a consequence

[*]A preliminary version of this paper appeared in the Proceedings of the 23rd IEEE Conference on Computational Complexity, 2008.

[†]Supported in part by NSF Grants CCF-0514155, DMS-0652582, CCF-0830133, and CCF-0832787.

[‡]Part of this work was done while visiting DIMACS and Rutgers University partially supported by NSF grant CCF-0514703. Supported in part by grant GA ČR 201/07/P276, project No. 1M0021620808 of MŠMT ČR and Institutional Research Plan No. AV0Z10190503.

of some very modest-sounding lower bound results (such as a lower bound of size $n^{1.0001}$). Of course, a confirmed pessimist would say that this is merely evidence that even these modest-sounding lower bounds are likely to remain beyond our reach. In Section 6 we discuss some possible interpretations of our results; in particular, we discuss the extent to which it might be possible to hope that the observations we present here point to a path around the obstacles to proving circuit lower bounds that were presented by Razborov and Rudich in their work on Natural Proofs [26].

1.1 Circuit complexity classes

This paper focuses on NC^1 and its subclasses. Let us remind the reader of the main definitions, and present some notation. For more background on circuit complexity, the reader is referred to the text by Vollmer [32].

- NC^1 is the class of languages recognized by circuits of fan-in two AND and OR gates, and unary NOT gates, having depth $O(\log n)$. Two standard complete problems for NC^1 are (1) the word problem for the permutation group S_5 on five elements [5], and (2) the Boolean Formula Evaluation problem [9]. In order to make the statement of some of our results slightly more crisp, we will be somewhat particular about the encoding of the Boolean Formula Evaluation problem. Define BFE to be the set of all *balanced* Boolean formulae (with constants 0 and 1, and no variables) that evaluate to 1, where the set of connectives is $\{\text{AND}, \text{OR}, \oplus\}$. This encoding of BFE remains complete for NC^1. (See, for example, the proof of Lemma 7.2 in [6].)

  We also make use of an NC^1-complete variant of s-t-connectivity. We say that a directed graph *is of width* k if it is a layered graph where each layer is of size at most k, the layers are ordered and every edge goes from vertices of one layer to the vertices of the next layer. W5-STCONN is the problem of deciding whether the first vertex of the first layer is connected by a path to the last vertex in the last layer of a width 5 directed graph. It follows from [5] that W5-STCONN is complete for NC^1. (The problem W5-STCONN remains complete for NC^1 if directed edges are permitted *in both directions* between adjacent layers, as well as in the undirected case. These variants of W5-STCONN could also be used for our discussion here, with minor technical modifications.)

- TC^0 is the class of languages recognized by polynomial-size constant-depth circuits of (unbounded fan-in) MAJ gates and unary NOT gates. (A MAJ gate is a gate that evaluates to one iff the majority of its inputs is set to one.)

- ACC^0 is the union of all the classes AC$^0[q]$ (for $q > 1$); see below.

- CC^0 is the union of all the classes CC$^0[q]$ (for $q > 1$); see below.

- AC$^0[q]$ is the class of languages recognized by polynomial-size constant-depth circuits of unbounded fan-in AND and OR gates and unary NOT gates, along with unbounded fan-in MOD-q gates. (A MOD-q gate evaluates to one iff the number of ones that feed into it is divisible by q).

- CC$^0[q]$ is the class of languages recognized by polynomial-size constant-depth circuits having *only* MOD-q gates.

- AC^0 is the class of languages recognized by polynomial-size constant-depth circuits of unbounded fan-in AND and OR gates and unary NOT gates.

As presented, these classes are *nonuniform* (i.e., it is not required that there be an easy way to construct the circuits for inputs of length n). We shall also need to consider logspace-uniform and Dlogtime-uniform versions of these classes [6].

Lower bounds are known for AC$^0[q]$ when q is prime [29], but it remains unknown even whether NP $=$ Dlogtime-uniform CC$^0[6]$.

1.2 What are the main contributions?

In Section 3 we show that many problems (such as BFE, W5-STCONN, the word problem over S_5, MAJ, AND, and iterated matrix product) have strong self-reducibility properties. Then, in Section 4, we show that, for any set possessing such self-reducibility properties, a lower bound of size n^c implies a superpolynomial size lower bound. (The constant "c" depends on the details of the self-reduction. For the word problem over S_5 or any of the problems BFE, W5-STCONN, MAJ or AND, any constant $c > 1$ suffices.)[1]

This seems to be a new observation. There are several examples of nonlinear lower bounds for various models of computation. For example Håstad presents a nearly-cubic lower bound on the formula size for a certain function [16], lower bounds on branching program size have been presented [1, 7], and the time-space tradeoff results that are surveyed by van Melkebeek [31] give run-time lower bounds of the form n^c for small-space computations. None of these lower bounds has led to separations of complexity classes. More to the point, there has never been any expectation that a lower bound of the form n^c could *possibly* lead to a separation of complexity classes. In this paper, we show that there are several settings where this *can* occur. Moreover, in Section 6 we show that the work of Razborov and Rudich on "Natural Proofs" [26] poses no barrier to proving weak lower bounds of the form n^c. This gives rise to some hope of separating circuit classes by proving circuit lower bounds using "natural" proof techniques.

It is necessary to be precise about the meaning of the word "size". There are two popular measures of circuit size—the number of gates and the number of wires. (There are always at least as many wires as there are gates. See e.g. [24] for treatment of the difference between these two measures of "size". For the results that have been mentioned in the paper thus far, the correct interpretation of "size" is "number of wires".) We will have occasion to refer to each of these two size measures, and in those cases where it is important to know which size measure is meant, we will be specific.

As mentioned above, in order to show that $\mathrm{TC}^0 \neq \mathrm{NC}^1$, it suffices to show that BFE requires TC^0 circuits of size $n^{1+\epsilon}$ for some constant $\epsilon > 0$. In fact, some non-linear lower bounds for BFE *are* known; Impagliazzo, Paturi, and Saks showed that any depth d TC^0 circuit for PARITY must have $n^{1+\Omega(1/(2.5)^d)}$ wires [20]. Since there is a trivial reduction from PARITY to BFE, the same size lower bound holds for BFE. Clearly, no proof of $\mathrm{TC}^0 \neq \mathrm{NC}^1$ can follow from a PARITY lower bound, and equally clearly, this argument does not yield a lower bound on the size of $\mathrm{AC}^0[6]$ circuits computing BFE. In fact, there seem to be no known lower bounds for BFE on $\mathrm{AC}^0[q]$ circuits for any composite q.

Fortnow showed that SAT does not have logspace-uniform NC^1 circuits of size $n^{1+o(1)}$ [13]. Since we are able to show that modest lower bounds for BFE would yield superpolynomial lower bounds, it is natural to wonder if the same situation holds for SAT. That is, if one could build on the Fortnow lower bound, and show that SAT requires $\mathrm{AC}^0[6]$ circuits of size $n^{1.01}$, would it follow that $\mathrm{NP} \neq \mathrm{AC}^0[6]$? We know of no such implication – and in Section 5 we show that the approach that works for BFE cannot transfer directly to SAT. More specifically, in Section 5 we show that any set possessing the self-reducibility properties that we utilize in Section 4 must lie in (uniform) NC. Thus, in order to demonstrate that SAT has the sort of self-reducibility properties that would enable us to amplify modest lower bounds to superpolynomial lower bounds, one would have to first prove that P=NP. (It is still conceivable that one could proceed by arguing that *if* $\mathrm{NP} = \mathrm{AC}^0[6]$, *then* SAT has the desired type of self-reduction, but we have not been able to construct such an argument.) It is interesting to note that Srinivasan has shown [30] that an $\Omega(n^{1+\epsilon})$ lower bound on the running time of algorithms that compute weak approximations to MAX-CLIQUE would imply $\mathrm{P} \neq \mathrm{NP}$. Using his techniques, we show in Section 7 that if $\mathrm{NP} = \mathrm{AC}^0[6]$, then there are $\mathrm{AC}^0[6]$ circuits of size $n^{1+o(1)}$ that compute $n^{1-o(1)}$-approximations to MAX-CLIQUE. This also yields lower bounds on the difficulty of reducing MAX-CLIQUE to approximations of MAX-CLIQUE.

[1]A special case of this general observation (relating only to regular sets) also appears in a survey article by the second author [23]; the present article expands significantly on the related results of [23].

Even though we do not know how to separate NP from AC$^0[6]$ by presenting a lower bound of the form n^c for the size of AC$^0[6]$ circuits for SAT, we would nonetheless like to be able to present such a lower bound (as an illustration that current techniques can provide the sort of modest lower bounds that would separate NC^1 from AC$^0[6]$ if such bounds could be proved for BFE). Although we can not provide such a lower bound, in Section 8 we do provide a lower bound analogous to the Impagliazzo, Paturi, and Saks bound mentioned above; we show that there is a constant c_d such that depth d AC$^0[6]$ circuits for SAT require size n^{1+c_d}.

2 Preliminaries

We have presented definitions for several constant-depth circuit complexity classes in Section 1.1. For any of these classes $\mathcal{C}$, we can also define $\mathcal{C}$-reducibility. We say that $A \leq^{\mathcal{C}}_{T} B$ if there is a constant-depth family of circuits of polynomial size recognizing A, where the circuits have *oracle gates* for the language B in addition to the collection of gates that is provided in the definition of the circuit class $\mathcal{C}$.

A $\mathcal{C}$ *self-reduction for* A is a family of oracle circuits witnessing that $A \leq^{\mathcal{C}}_{T} A$, where on input x, the oracle circuit does not feed input x into any of its oracle gates.

A *pure self-reduction for* A is a self-reduction for A, where the *only* gates are oracle gates, as well as bounded fan-in AND and OR gates and unary NOT gates.[2]

Self-reductions can be either uniform or non-uniform. The reader can verify that all of the examples of self-reductions that we present in this paper are Dlogtime-uniform.

In addition to languages over the binary alphabet, we also consider languages over an arbitrary alphabet Σ. In such cases we assume that there is some fixed encoding of symbols from Σ into fixed-length binary strings; circuits for languages in Σ^* operate on these Boolean encodings. Similarly, a circuit for a function with non-Boolean output produces a binary encoding of the output symbol.

3 Downward self-reducibility

Definition 1 *Let $f : \{0,1\}^* \rightarrow \{0,1\}^*$ be a function. Let $s(n), m(n) : \mathbb{N} \rightarrow \mathbb{N}$ be functions such that for all n, $m(n) < n$ and let $d \geq 1$ be an integer. We say that f_n* is downward self-reducible to $f_{m(n)}$ by a pure reduction of depth d and size $s(n)$ *if for every n there exists a depth d pure self-reduction with $s(n)$ gates computing f_n, using oracle gates only for $f_{m(n)}$.*

Similarly, we can write of f_n being downward self-reducible to $f_{m(n)}$ by a $\mathcal{C}$ reduction of depth d and size $s(n)$ for various circuit classes $\mathcal{C}$. This notion of downward self-reducibility is essentially identical to what Goldwasser *et al.* call “strong downward self-reducibility” [15]. For our purposes, it is important to pay close attention to the size and depth of the reduction.

The following example may seem trivial, but it is nonetheless useful.

Proposition 2 *For any $0 < \epsilon < 1$, AND_n is downward self-reducible to AND_{n^ϵ} by a pure reduction of depth $O(1/\epsilon)$ and size $O(n^{1-\epsilon})$. Similarly for OR_n.*

Proof. Form a tree of depth $1/\epsilon$ from gates computing AND_{n^ϵ} and assign each input bit to one of the leaves. Clearly, the circuit will compute AND_n and it consists of $O(n^{1-\epsilon})$ gates. □

The case of AND and OR can be further generalized as follows. Let M be a finite monoid (a finite set with an associative binary operation and identity element.) We denote the operation of M multiplicatively. The word problem over M is the function $W_M : \{0,1\}^* \rightarrow \{0,1\}^{|M|}$ that takes binary encodings of several elements from M and outputs the binary encoding of their product. (The

[2]One could perhaps call pure self-reductions “NC^0 self-reductions”, but since the oracle gates have unbounded fan-in, this seems to be quite different than NC^0 computation.

particular way of encoding elements from M into binary representation is of no interest to us. We may assume that it is the unary encoding: $1^i 0^{|M|-i}$ denoting the i-th element of M.)

Proposition 3 *For any monoid M and any $0 < \epsilon < 1$, $(W_M)_n$ is downward self-reducible to $(W_M)_{n^\epsilon}$ by a pure reduction of depth $O(1/\epsilon)$ and size $O(n^{1-\epsilon})$.*

The proof is essentially the same as for AND and OR. If for an integer $q > 1$ we consider the monoid $(\{0, 1, \ldots, q-1\}, +(\bmod q))$ then we obtain the next corollary.

Corollary 4 *For any $0 < \epsilon < 1$, $(\text{MOD}-q)_n$ is downward self-reducible to $(\text{MOD}-q)_{n^\epsilon}$ by a pure reduction of depth $O(1/\epsilon)$ and size $O(n^{1-\epsilon})$.*

A similar proof also yields:

Proposition 5 *For any $0 < \epsilon < 1$, W5-STCONN$_n$ is downward self-reducible to W5-STCONN$_{n^\epsilon}$ by a pure reduction of depth $O(1/\epsilon)$ and size $O(n^{1-\epsilon})$.*

We can prove a similar claim also for MAJ. This time the proof is a little bit more involved and uses the following lemma.

Lemma 6 *For any $m, \ell \geq 1$ there is a constant depth circuit with $O(m \log m)$ oracle gates for MAJ_{2m} in addition to bounded fan-in AND and OR gates and unary NOT gates, taking as its input $m \times \ell$ bits representing m ℓ-bit integers, and producing as output a sequence of ℓ integers (each of $\ell + \log m$ bits) that has the same sum as the input integers.*

Proof. First, observe that we can compute AND_m and OR_m, using a gate for MAJ_{2m} and constants 0 and 1.

Using m gates for MAJ_{2m} (together with some AND_m and OR_m gates that can be computed with MAJ_{2m}), we can compute the unary representation of the sum of m bits (i.e., $1^i 0^{m-i}$ where i of the input bits are 1). This unary representation can be further transformed into binary representation by a constant depth circuit using $O(m \log m)$ AND_m, OR_m and NOT gates. Thus we can sum the input bits at each of the ℓ binary positions in the m input numbers, to obtain ℓ $\ell + \log m$-bit integers representing the sum of the input. (Note each of these ℓ integers will have ℓ of its bits always set to zero.) □

Proposition 7 *For any $0 < \epsilon < 1$, MAJ_n is downward self-reducible to MAJ_{n^ϵ} by a pure reduction of depth $O(1/\epsilon)$ and size $O(n \log n)$.*

Proof. We prove the claim for $\epsilon = 1/2$. For smaller ϵ the proof follows using the same technique of building a tree as in the previous propositions. We can view the input as n 1-bit integers. To determine the output of MAJ_n we will compute the binary representation of the sum of these integers. We proceed in summing them as follows. We split the input into $2\sqrt{n}$ blocks of $\sqrt{n}/2$ input bits, each representing $\sqrt{n}/2$ 1-bit integers. By the preceding lemma we can obtain the sum of each block using $O(\sqrt{n} \log n)$ $\text{MAJ}_{\sqrt{n}}$ gates, i.e., $O(n \log n)$ $\text{MAJ}_{\sqrt{n}}$ gates in total.

Hence we have reduced the problem of summing the input bits to the problem of summing $2\sqrt{n}$ $O(\log n)$-bit integers. Splitting the integers into four equal size groups and applying the lemma on each of the groups gives $O(\log n)$ $O(\log n)$-bit integers whose sum is equal to the input sum.

We divide each of these integers into blocks of $\log \log n$ consecutive bits and we sum the corresponding blocks from the $O(\log n)$ integers using the lemma. For each block this yields $O(\log \log n)$ integers, each having $O(\log \log n)$ bits, which sum to the sum of the block. Furthermore, by a DNF formula of size $2^{O(\log \log n)^2} \leq n^{o(1)}$ built from $\text{AND}_{O((\log \log n)^2)}$ and $\text{OR}_{n^{o(1)}}$ gates we can obtain for each block its $O(\log \log n)$-bit sum. From these $O(\log n / \log \log n)$ $O(\log \log n)$-bit sums we can form $O(1)$ $O(\log n)$-bit integers that represent the sum of the input bits. Summing $O(1)$ $O(\log n)$-bit integers can be done using $O(\log^3 n)$ $\text{AND}_{O(\log n)}$ and $\text{OR}_{O(\log n)}$ gates; this concludes the proof. $\square$

We have seen that AND, OR, MOD-q, and MAJ are all downward self-reducible. We saw also that downward self-reducibility holds for the word problem over any finite monoid, which yields self-reductions for some of the standard complete problems for NC^1: W5-STCONN and the word problem over S_5. We thank Mario Szegedy for pointing out that BFE (another standard complete problem for NC^1) is also downward self-reducible:

Proposition 8 *For any $0 < \epsilon < 1$, BFE_n is downward self-reducible to BFE_{n^ϵ} by a pure reduction of depth $O(1/\epsilon)$ and size $O(n)$.*

Proof. Since the input is a balanced formula of size n, the depth of the formula is $\log n$. We can cut this formula into $1/\epsilon$ layers, each of depth $\epsilon \log n$. We will evaluate the formula, starting with the subformulae whose roots are on the top of the bottom layer (whose inputs are the leaves of the original formula). Each of these formulae has size n^ϵ. We feed the values for each of those subformulae into the formulae that form the next layer, and so on. $\square$

Indeed, we point out that any problem complete for a complexity class that has a downward self-reducible complete problem must be downward self-reducible. See Proposition 19 in the next section.

Another problem for which we can prove downward self-reducibility is *Iterated Matrix Multiplication*. Let $\text{IMM}_{n,d,\ell} : \{0,1\}^{nd^2\ell} \to \{0,1\}^{d^2 n(\ell+\log d)}$ be the problem of computing the product of n $d \times d$ matrices, with each entry being a non-negative ℓ-bit integer. Define the *modular* version of the Iterated Matrix Product to be the function $\text{mIMM}_{n,d,q} : \{0,1\}^{nd^2 \log q} \to \{0,1\}^{d^2 \log q}$ computing the Iterated Matrix Product modulo some integer $q \geq 2$. Finally, we will also need to consider the *Boolean* Iterated Matrix Product problem $\text{BIMM}_{n,d} : \{0,1\}^{nd^2} \to \{0,1\}^{d^2}$ which is the Iterated Matrix Problem over the ring $(\{0,1\}, \text{OR}, \text{AND})$.

The following proposition is immediate:

Proposition 9 *For any $0 < \epsilon < 1$ and any $n, d, q \geq 1$, $\text{mIMM}_{n,d,q}$ is downward self-reducible to $\text{mIMM}_{n^\epsilon,d,q}$ by a pure reduction of depth $O(1/\epsilon)$ and size $O(n^{1-\epsilon})$. $\text{BIMM}_{n,d}$ is similarly reducible to $\text{BIMM}_{n^\epsilon,d}$ with the same parameters.*

The following more interesting lemma will be useful in the next section.

Lemma 10 *There is a universal constant c_{CRR} such that for any $0 < \epsilon < 1$ and any $d \leq n$ (where $d = d(n)$ may be a function of n), $\text{IMM}_{n,d,n}$ is downward self-reducible to $\text{IMM}_{n^\epsilon,d,n^\epsilon}$ by a TC^0-reduction of depth $O(1/\epsilon)$, with $O(d^2 \cdot n^{3+2c_{\text{CRR}}})$ wires and $O(n^{3-\epsilon})$ oracle gates.*

Here, c_{CRR} is a specific constant that can be determined from a paper of Hesse et al. [19].
Proof. Hesse et al. [19] give uniform TC^0 circuits with $O(n^{c_{\text{CRR}}})$ wires that do the following tasks:

- take as input two n-bit integers a and b, and output $a \bmod b$.
- take as input an n-bit integer a, and output its *Chinese Remainder Representation*, i.e., a sequence of $O(n)$ pairs (a_i, b_i) of $O(\log n)$-bit numbers where $a_i = a \bmod b_i$ and all b_i are distinct primes.
- take as input n pairs (a_i, b_i) of $O(\log n)$-bit numbers and output an $O(n \log n)$-bit number a satisfying $a_i = a \bmod b_i$ and $0 \leq a < \prod_i b_i$, if the b_i are distinct primes.

Using these circuits we can reduce $\text{IMM}_{n,d,n}$ to the problem of computing $O(n^2)$ instances of mIMM_{n,d,q_i} in parallel for $O(n^2)$ distinct prime $O(\log n)$-bit numbers q_i. Namely to compute the iterated product, we first compute the representation of each input matrix mod each of the primes q_i

(thereby converting the input from binary representation to Chinese Remainder Representation); this gives us $O(n^2)$ instances of mIMM$_{n,d,q_i}$ to solve. Next, we compute the iterated product mod each of the q_i (thereby obtaining the output in Chinese Remainder Representation). Finally, we convert the answer to binary representation.

By the previous proposition, for each i we can downward reduce the computation of mIMM$_{n,d,q_i}$ to mIMM$_{n^\epsilon,d,q_i}$. However, since our goal is to produce a self-reduction for IMM, we must show how to simulate each call to mIMM using an oracle for IMM. But this is easy: if inputs to mIMM are fed instead into a IMM gate, then by taking the output from the IMM gate and taking each entry mod q_i, we obtain the output that would have been given by the mIMM gate. That is, we use TC^0 circuitry to prepare the inputs that would (ideally) be presented to the mIMM$_{n^\epsilon,d,q_i}$ oracle gates, and instead we use IMM$_{n^\epsilon,d,n^\epsilon}$ gates (which provide the correct answer mod q_i.) We then again use TC^0 circuitry to take each matrix entry mod q_i, thereby simulating one oracle gate in a mIMM self-reduction.

The size of the resulting circuit is going to be

- $d^2 n \cdot O(n^{2c_{\mathrm{CRR}}})$ to convert the input into Chinese Remainder Representation relative to $O(n^2)$ moduli and then convert back from Chinese Remainder Representation into binary, plus
- $O(n^2 \cdot n^{1-\epsilon} \cdot d^2 \cdot n^{2\epsilon c_{\mathrm{CRR}}})$ for taking remainders to process the output of the $O(n^2 \cdot n^{1-\epsilon})$ oracle gates.

Hence we get a TC^0 circuit reducing IMM$_{n,d,n}$ to IMM$_{n^\epsilon,d,n^\epsilon}$ of size $O(d^2 \cdot n^{3+2c_{\mathrm{CRR}}})$. □

4 Amplifying lower bounds

In the previous section we have established several downward self-reducibility results. In this section we show that any problem that is downward self-reducible in this way has circuits of polynomial size if and only if it has very small circuits. Thus, if a small circuit size lower bound can be proved for any such problem, it can be "amplified" into a superpolynomial size lower bound.

The general form of our claims is:

> If a function f is computable by polynomial size circuits of type $\mathcal{C}$ then for any $\epsilon > 0$, f is computable by circuits of type $\mathcal{C}$ using $O(n^{1+\epsilon})$ gates and wires.

The circuit types we will consider are AC^0, ACC^0, CC^0, TC^0 and NC^1 circuits. The functions f we will consider will typically (but not always) be complete for some complexity class. For example MAJ is complete for TC^0 (under $\leq_T^{\mathrm{AC}^0}$ reductions), and the word problem for S_5 is complete for NC^1, and so on. The consequence of our claim is that establishing a lower bound of $\Omega(n^{1+\epsilon})$ for some $\epsilon > 0$ on the number of wires or gates necessary to compute f would separate some of the circuit classes. The following proposition summarizes known relationships between these circuit classes.

Proposition 11

$$\mathrm{AC}^0 \subsetneq \mathrm{ACC}^0 \subseteq \mathrm{TC}^0 \subseteq \mathrm{NC}^1$$

$$\mathrm{CC}^0 \subseteq \mathrm{ACC}^0, \;\; \mathrm{CC}^0 \not\subseteq \mathrm{AC}^0$$

Except for the proper inclusion AC^0 $\subsetneq$ ACC^0 [14, 33, 18] which also implies CC^0 $\not\subseteq$ AC^0 the precise relationship among ACC^0, CC^0, TC^0 and NC^1 is not known, and any separation or collapse would constitute major progress in theoretical computer science. Separation of, say, TC^0 from NC^1 would typically entail showing that no polynomial size TC^0 circuit can compute some chosen function from NC^1. We show that a weaker lower bound than super-polynomial can already yield the same conclusion.

Theorem 12 *If, for every $\epsilon > 0$, f_n is downward self-reducible to f_{n^ϵ} by a pure reduction of depth $O(1/\epsilon)$ and size $s(n)$, and $f \in \mathcal{C}$, then for every $\epsilon' > 0$, f has circuits of type $\mathcal{C}$ with $O(s(n)n^{\epsilon'})$ wires.*

Proof. Assume that f_n has circuits of type $\mathcal{C}$ with n^k wires. The reduction of f_n to f_{n^ϵ} has at most $s(n)$ oracle gates, each of fan-in n^ϵ, and at most $s(n)$ other gates of bounded fan-in. Thus the total number of wires in the reduction is $O(s(n)n^\epsilon)$. If we replace each oracle gate for f_{n^ϵ} by the circuit of type $\mathcal{C}$ of size $n^{\epsilon k}$, we obtain a circuit of type $\mathcal{C}$ for f_n with $O(s(n)n^\epsilon n^{\epsilon k}) = O(s(n)n^{\epsilon(k+1)})$ wires. The claim follows, because k is fixed and the hypothesis holds for every $\epsilon > 0$. $\square$

In the previous theorem, note that if $\mathcal{C}$ is a class of *bounded depth* circuits, then f has circuits of type $\mathcal{C}$ having depth $O(1/\epsilon')$ and $O(s(n)n^{\epsilon'})$ wires. For most of our arguments, $s(n) = O(n \log n)$. This yields the following corollary.

Corollary 13

1. *If for some $\epsilon > 0$,* W5-STCONN *requires* CC^0 *circuits with at least $\Omega(n^{1+\epsilon})$ wires, then* $\mathrm{CC}^0 \neq \mathrm{NC}^1$. *The same is true for* ACC^0 *and* TC^0 *in place of* CC^0, *and for* BFE *and W_{S_5} in place of* W5-STCONN.
2. *If for some $\epsilon > 0$,* MAJ *requires* CC^0 *circuits with at least $\Omega(n^{1+\epsilon})$ wires (gates) then* $\mathrm{CC}^0 \neq \mathrm{TC}^0$. *The same is true for* ACC^0 *in place of* CC^0.
3. *If for some $\epsilon > 0$,* AND *requires* CC^0 *circuits with at least $\Omega(n^{1+\epsilon})$ wires (gates) then* $\mathrm{CC}^0 \neq \mathrm{ACC}^0$.

Contrast this with the situation for SAT; if SAT is in TC^0, we have no way to bound the number k such that TC^0 size n^k is sufficient to compute SAT. (Although, as we mentioned in Section 1.2, Srinivasan has shown that if P $=$ NP then there are algorithms running in time $n^{1+\epsilon}$ that compute *weak approximations* to MAX-CLIQUE [30]. See also our Section 7.)

Although stated as a sequence of implications, the preceding corollary is really a sequence of *equivalences*, since W5-STCONN is complete for NC^1, MAJ is complete for TC^0, and AND is complete for ACC^0 under $\leq_T^{\mathrm{CC}^0}$ reductions. Thus, for example, W5-STCONN is in ACC^0 iff $\mathrm{NC}^1 = \mathrm{ACC}^0$.

We remark that, since our self-reductions are Dlogtime-uniform, one can compute a constant K such that, for example, if BFE is in Dlogtime-uniform TC^0, then it has TC^0 circuits with $O(n^{1+\epsilon})$ wires where the uniformity machine runs in time $K \log n$. (We have not computed the value of K – and indeed this value may depend on minor details of the particular formulation that is used in defining Dlogtime-uniformity – but we anticipate that $K = 4$ is sufficient; the self-reductions have a *very* regular structure, and the $O(\log n)$ running time of the "original" TC^0 circuit family ends up being simulated only to determine the structure of circuits for inputs of size n^ϵ for small values of ϵ.)

Sometimes concrete lower bounds are easier to prove for specially-constructed sets, rather than for the standard complete sets for a complexity class. The following corollary shows that we can also "amplify" lower bounds for such specially-constructed sets, since if one can show that a specially-constructed set lies in NC^1, then typically one can determine some upper bound on the depth $d(n)$ of the NC^1 circuits computing f.

Corollary 14 *Let f be computable by* NC^1 *circuits of depth $d(n)$. If f does not have* TC^0 *circuits of size $O(3^{d(n)})$ then* $\mathrm{TC}^0 \neq \mathrm{NC}^1$. *Similarly for* ACC^0 *and* CC^0 *in place of* TC^0.

Proof. If f has NC^1 circuits of depth $d(n)$, then it has a balanced formula of size $2^{d(n)}$, and thus there is a reduction of f to instances of BFE of size $2^{d(n)}$. If $\mathrm{TC}^0 = \mathrm{NC}^1$ then evaluating Boolean formulae of length ℓ can be done by TC^0 circuits of size $O(\ell^{1+\epsilon})$ for any chosen $\epsilon > 0$. The claim follows. $\square$

The technique is applicable also to other circuit classes, so if we pick a function f from e.g. TC^0 and we know that it is computable by TC^0 circuits of size $O(n^k)$, then if $\mathrm{TC}^0 = \mathrm{ACC}^0$ then for every

$\epsilon > 0$, f is computable by ACC^0 circuits using $O(n^{k(1+\epsilon)})$ wires (gates). So proving an $\Omega(n^{k(1+\epsilon)})$ lower bound on the size of ACC^0 circuits for f separates ACC^0 from TC^0.

This technique is applicable, to a certain extent, also to classes larger than NC^1. First, let us consider NL. Boolean iterated matrix product BIMM$_{n,n}$ is complete for NL. We do not know how to work directly with BIMM$_{n,n}$, and thus we work with slightly smaller matrices instead.

Theorem 15 *If* NL $\subseteq$ NC^1 *then* BIMM$_{n,2^{\sqrt{\log n}}}$ *is computable by* NC^1 *circuits with* $o(n^2)$ *wires. The same is true for* CC^0, ACC^0, *and* TC^0 *in place of* NC^1.

(The contrapositive may be more informative; if one can show that BIMM$_{n,2^{\sqrt{\log n}}}$ requires NC^1 circuits of size $\Omega(n^2)$ then one has shown that NC^1 $\neq$ NL. Unlike the earlier theorems in this section, we obtain only an implication, and not an equivalence – since BIMM$_{n,2^{\sqrt{\log n}}}$ is not known (or believed) to be complete for NL. Note that this result is for NC^1 *circuit* size; it does not seem to translate into a useful statement about *formula* size.)

Proof. Since BIMM$_{n,n}$ is in NL, our assumption implies that BIMM$_{n,n}$ is computable by NC^1 circuits of size $O(n^k)$ for some $k > 0$. Choose $\epsilon = 1/k$. Then BIMM$_{n^\epsilon,n^\epsilon}$ is computable by NC^1 circuits of size $O(n^{\epsilon k}) = O(n)$ and hence BIMM$_{n^\epsilon,2^{\sqrt{\log n}}}$ is computable by NC^1 circuits of size $O(n)$. By Proposition 9, BIMM$_{n,2^{\sqrt{\log n}}}$ is downward self-reducible to BIMM$_{n^\epsilon,2^{\sqrt{\log n}}}$ by a pure reduction of size $n^{1-\epsilon}$. The number of wires in this reduction is $n^{1-\epsilon} \cdot n^\epsilon 2^{2\sqrt{\log n}} = n2^{2\sqrt{\log n}}$. Since BIMM$_{n^\epsilon,2^{\sqrt{\log n}}}$ has NC^1 circuits of size $O(n)$, we can replace each oracle gate by a circuit with $O(n)$ wires, yielding an NC^1 circuit with $O(n2^{2\sqrt{\log n}} + n^{1-\epsilon}n) = o(n^2)$ wires. □

We now turn to the complexity class #L (the class of functions that count the number of accepting paths of NL machines). This is the largest complexity class that we know how to address using these techniques. Iterated Matrix Multiplication IMM$_{n,n,n}$ is a problem complete for #L (see [4]). IMM$_{n,2^{\sqrt{\log n}},n}$ is a subproblem not known (or expected) to be complete for #L, but also not known to lie in any smaller complexity class.

Theorem 16 *If* #L $\subseteq$ TC^0 *then* IMM$_{n,2^{\sqrt{\log n}},n}$ *is computable by* TC^0 *circuits with* $O(n^{2c_{\mathrm{CRR}}+4})$ *wires. Similarly if* #L $\subseteq$ NC^1 *then* IMM$_{n,2^{\sqrt{\log n}},n}$ *is computable by* NC^1 *circuits of size* $O(n^{4c_{\mathrm{CRR}}+8})$ *wires.*

Thus to separate #L from TC^0 it suffices to show a lower bound of $\omega(n^{2c_{\mathrm{CRR}}+4})$ on the size of TC^0 circuits computing IMM$_{n,2^{\sqrt{\log n}},n}$. Similarly for NC^1.

Proof. Since IMM$_{n,n,n}$ is in #L, by our assumption, IMM$_{n,n,n}$ is computable by TC^0 circuits of size $O(n^k)$ for some $k > 0$. Choose $\epsilon = 1/k$. Then IMM$_{n^\epsilon,n^\epsilon,n^\epsilon}$ is computable by TC^0 circuits of size $O(n^{\epsilon k}) = O(n)$ and hence IMM$_{n^\epsilon,2^{\sqrt{\log n}},n^\epsilon}$ is computable by TC^0 circuits of size $O(n)$.

By Lemma 10, IMM$_{n,2^{\sqrt{\log n}},n}$ is downward self-reducible to IMM$_{n^\epsilon,2^{\sqrt{\log n}},n^\epsilon}$ by TC^0 circuits of size $O(2^{O(\sqrt{\log n})} \cdot n^{2c_{\mathrm{CRR}}+3}) = O(n^{2c_{\mathrm{CRR}}+4})$. There are $O(n^{3-\epsilon})$ oracle gates in this reduction, and each gate for IMM$_{n^\epsilon,2^{\sqrt{\log n}},n^\epsilon}$ can be replaced by circuits with $O(n)$ wires, yielding TC^0 circuits of size $O(n^{2c_{\mathrm{CRR}}+4} + n^4) = O(n^{2c_{\mathrm{CRR}}+4})$. This yields the bound for TC^0 circuits in the statement of the lemma.

To prove the second claim in the theorem, regarding NC^1 circuits, it suffices to remark that each MAJ$_n$ gate can be replaced by NC^1 circuitry, at most squaring the size. (Tighter analysis is possible.) □

The preceding two theorems do not make use of problems that are known to be *complete* for well-known complexity classes, and thus we obtain only *implications* regarding NL and #L, instead of *equivalent* statements concerning whether these classes collapse with NC^1. However, it is worthwhile noting that IMM$_{n,3,n}$ is complete for GapNC^1 [10] (the class of functions over the integers, computable by polynomial-size arithmetic formulae). All functions in NC^1 are in GapNC^1, and it has been conjectured that GapNC^1 coincides with NC^1 [2]. GapNC^1 is the only well-studied complexity class not known to be contained in NC^1, for which we can present a complete problem that is strongly downward self-reducible. The proof of the preceding theorem yields the following pair of equivalences.

Theorem 17
- *GapNC*$^1 \subseteq \mathrm{NC}^1$ *iff* $\mathrm{IMM}_{n,3,n}$ *has* NC^1 *circuits of size* $n^{3+2c_{\mathrm{CRR}}}$.
- *GapNC*$^1 \subseteq \mathrm{TC}^0$ *iff* $\mathrm{IMM}_{n,3,n}$ *has* TC^0 *circuits of size* $n^{3+2c_{\mathrm{CRR}}}$.

5 Limits on downward self-reducibility

In the previous section we have seen that downward self-reducibility provides us with an interesting tool for the study of circuit classes. We have shown that in order to separate circuit classes such as ACC^0 and NC^1, quadratic lower bounds for the circuit complexity of certain NC^1-complete problems would suffice. What about separating ACC^0 from, say NP? That should in principle be a much easier task. Can we use the technique of downward self-reducibility to establish an analog of Corollary 13 for ACC^0 versus NP?

The following theorem shows that there are significant obstacles to overcome before such an approach can work. Namely, in order to establish that a problem is downward self-reducible in the way that we study in Section 3, one must already have an efficient algorithm for the problem.

Theorem 18 *Let* $f : \{0,1\}^* \to \{0,1\}^*$ *be a function, and* $m(n) : \mathbb{N} \to \mathbb{N}$ *be such that* $m(n) < n^\epsilon$ *for some* $0 < \epsilon < 1$ *and all* $n \geq 2$.

1. *If* f_n *is downward self-reducible to* $f_{m(n)}$ *by* TC^0*-reductions, then* $f \in \mathrm{NC}$ *and has* TC^0 *circuits of size* 2^{n^δ} *for every* $\delta > 0$.
2. *If* f_n *is downward self-reducible to* $f_{m(n)}$ *via polynomial time Turing reductions then* f *is in* P.

Proof. 1) In order to build a circuit for f_n, start with the TC^0 circuit of depth d and size n^k that reduces f_n to $f_{m(n)}$. If we replace each oracle gate in this circuit with the circuit that reduces $f_{m(n)}$ to $f_{m(m(n))}$, the depth of the new circuit is d^2 and the size is at most $n^k + n^k \cdot n^{\epsilon k}$. We repeat the process until the oracle gates are of size $O(1)$, at which point we replace the oracle gates by circuitry of size $O(1)$ computing f on small inputs. The number of stages is $O(\log\log n)$; thus the depth is $d^{O(\log\log n)} = \log^{O(1)} n$. The size of the circuit is bounded by $n^k \cdot n^{\epsilon k} \cdot n^{\epsilon^2 k} \cdots \leq n^{k/(1-\epsilon)}$. It is easy to verify that the resulting circuit is logspace-uniform if the self-reduction circuits are. This establishes that $f \in \mathrm{NC}$. In order to see that f has TC^0 circuits of size 2^{n^δ}, merely follow the same iteration process as above, but continue for only $O(1)$ stages instead of $O(\log\log n)$ stages. This results in a TC^0 oracle circuit with oracle gates for f_m with $m < n^\delta$. Now replace each oracle gate with a DNF expression for f_m. (Clearly, if the self-reduction is an AC^0 circuit instead of a TC^0 circuit, then f has AC^0 circuits of size 2^{n^δ}.)

2) Again we use the obvious recursive algorithm. We run the Turing reduction and whenever it asks an oracle query about a smaller instance of f we recursively invoke the reduction on the smaller instance. If the reduction runs in time $O(n^k)$ then the total running time of the algorithm will be bounded by $n^k \cdot n^{\epsilon k} \cdot n^{\epsilon^2 k} \cdots \leq n^{k/(1-\epsilon)}$. Since ϵ is constant, the time is polynomial. □

Speculation: Although Theorem 18 suggests that we abandon any attempt to show that SAT has the downward self-reducibility property, it does not exclude the following approach for trying to prove an analog of Corollary 13 for NP. Rather than trying to present a self-reduction for SAT *unconditionally*, perhaps one can start with the *assumption* that NP $\subseteq$ TC^0 and construct a downward self-reduction of SAT (or some other specially-constructed set in NP) and conclude that under this assumption SAT has almost linear size TC^0 circuits.

This is the appropriate time to observe that if NP $\subseteq$ TC^0, then it certainly does have the strong downward self-reducibility property; this follows from Proposition 19 below. However, since one can say nothing about the size of this self-reduction (other than that it is computed by an AC^0 circuit of polynomial size), this does not seem to allow us to conclude that SAT has TC^0 circuits of, say, quadratic size.

Proposition 19 *If A is equivalent to* BFE *under uniform (non-uniform, respectively) $\leq_T^{\mathrm{AC}^0}$ reductions, then for every $\epsilon > 0$, A_n is downward self-reducible via a uniform (non-uniform, respectively)* AC^0 *reduction of depth $O(1)$ and size $n^{O(1)}$ that asks queries of length at most n^ϵ. Moreover, the size of the self-reduction of A_n can be determined from the sizes of reductions between A and* BFE.

Proof. By hypothesis, $A \leq_T^{\mathrm{AC}^0}$BFE via a reduction that, on instances of length n, asks queries of size $n^{O(1)}$. Since queries to BFE can be padded easily to equivalent queries of longer length, we may assume that all queries have length n^k. Similarly, we are given that BFE$\leq_T^{\mathrm{AC}^0} A$ via a reduction that, on inputs of length m, asks queries of size at most m^c. Composing these reductions with the self-reduction that reduces BFE$_{n^k}$ to BFE$_{n^{k\delta}}$ (for $\delta < \epsilon/kc$) yields the desired self-reduction for A. □

The next section addresses the question of whether superpolynomial lower bounds obtained by "amplifying" a "natural" proof of a lower bound of size $n^{1.0001}$ would constitute an "*un-natural proof*".

6 The Natural Proofs barrier

Razborov and Rudich [26] identified a significant obstacle to further progress in proving lower bounds on circuit size, by observing that existing lower bound arguments rely on the existence of an easy-to-recognize combinatorial property of a function f that (a) is shared by a large fraction of all functions, and (b) is shared by no function that has small circuits of a given type. Razborov and Rudich showed that any "Natural Proof" that follows this paradigm and shows that a function cannot be computed by circuits of a class $\mathcal{C}$ constitutes a proof that $\mathcal{C}$ cannot compute pseudorandom function generators. It is not clear how significant an obstacle this is, for proving lower bounds against ACC^0, since there is not much evidence that ACC^0 circuit families can compute pseudorandom function generators. However, for TC^0 this is a serious impediment, since Naor and Reingold have presented a good candidate pseudorandom function generator that is computable in TC^0 [25].

It is premature to argue very strongly that we have identified a path around this obstacle. After all, the only new lower bound that this paper offers is to be found in Section 8, and that bound follows from known time-space tradeoff results. (These time-space tradeoffs, in turn, rely on diagonalization, which lies outside the natural proofs framework, but only gives lower bounds for *uniform* circuit families. The natural proofs framework addresses the problem of finding lower bounds for *nonuniform* circuit complexity.)

However, we contend that it is at least plausible that a natural proof could form the basis for a proof that NC$^1 \neq$ TC^0, even assuming that the Naor-Reingold generator is cryptographically secure.

How?

There seems to be no reason why a natural proof cannot yield a lower bound of the form n^k for some fixed k. The parity lower bound of Impagliazzo, Paturi, and Saks gives a lower bound of this form for BFE on TC^0 circuits of depth d [20]. Håstad gives a nearly cubic lower bound on formula size [16]. These are natural proofs.

The self-reducibility property that allows a modest lower bound to be amplified to a superpolynomial-size lower bound, on the other hand, is a combinatorial property that is shared by only a *vanishingly small fraction* of all Boolean functions on n variables. Thus, this part of a lower bound argument would *not* fit into the Natural Proofs framework. (Strictly speaking, the downward self-reducibility property is not a combinatorial property in the sense of the Natural Proofs framework, as it is a relationship between function values on different input sizes. However, all downward self-reducible functions must have truth-tables of small Kolmogorov complexity (since the truth-table of size 2^n is determined completely by a truth-table of size 2^{n^δ}), and thus they constitute a tiny fraction of all functions.)

To be concrete, let us exhibit an example of a property T that is *natural*, and *useful* in the sense of Razborov and Rudich. We will recall the definitions of Razborov and Rudich [26]:

Let F_n denote the class of all Boolean functions $f_n : \{0,1\}^n \rightarrow \{0,1\}$. A property $\{T_n \subseteq F_n\}_{n \in \mathbb{N}}$ is $QuasiP$-*natural* if there is a sub-property $\{T_n^* \subseteq T_n\}_{n \in \mathbb{N}}$ such that for some $\epsilon, c > 0$

1. $|T_n^*| \geq |F_n|/2^{\epsilon n}$, and
2. there is a deterministic algorithm that given a truth-table of a function $f_n : \{0,1\}^n \rightarrow \{0,1\}$ decides whether $f_n \in T_n^*$ in time 2^{n^c}.

Furthermore, a property $\{T_n \subseteq F_n\}_{n \in \mathbb{N}}$ is *useful* against a circuit class Λ if no sequence of functions $\{f_n \in T_n\}_{n \in \mathbb{N}}$ is computable by circuits from Λ.

Our property T is defined as follows:

$T_n = \{f_n : \{0,1\}^n \rightarrow \{0,1\};\ f_n$ does not have circuits of depth $\log^* n$ and size n^2 consisting of MAJ and NOT gates$\}$.

It is a trivial exercise to verify that T is natural and useful against TC^0 circuits of size $O(n^{1.5})$. Of course, we are not able to establish that BFE has property T; if it does, then by Corollary 13 NC^1 $\neq$ TC^0. Clearly, this argument makes use of no special properties of TC^0; one can easily come up with a $QuasiP$-natural property that will be useful against any class of circuits of a fixed polynomial size.

However, the existence of property T does not seem to imply anything very interesting about the nonexistence of pseudorandom function generators (and consequently does not yield interesting upper bounds on the complexity of factoring Blum integers, which would follow if the Naor-Reingold generator is insecure [25]).

The arguments of Razborov and Rudich transform any natural lower bound proof into a lower bound on the complexity of computing a pseudorandom function generator. However, lower bounds for circuits of size n^k for small fixed k translate into lower bounds for pseudorandom function generators that are so weak as to be uninformative.

So are there reasons to be more optimistic about prospects for lower bounds? We are not sure. The truth is that we do not understand computation. All the known lower bounds essentially rest on information theoretic arguments and none of them really takes into account *computation*. For example we are unable to handle *recursion* so our bounds typically deteriorate with depth. Hence, the underlying message of Razborov and Rudich – namely, that we need to go beyond combinatorial arguments – is still a worthwhile message. We identify two still unresolved challenges that we believe would advance our understanding of computation:

- Prove $\Omega(n^2)$ lower bounds on the length of width 5 branching programs computing an explicit function.
- Prove $\Omega(n^{1+1/\sqrt{d}})$ lower bounds on the size of depth d circuits computing an explicit function.

Are there perhaps fundamental barriers that remain in our path, as we attempt to prove circuit lower bounds?

One way to explore this question is to follow the lead of Razborov [28], who showed that (under cryptographic assumptions) the bounded arithmetic proof system S_2^2 cannot prove that SAT requires circuits of superpolynomial size. (In earlier work, Razborov had argued that most existing lower bound arguments can be carried out in even weaker systems [27].)

Perhaps techniques similar to those of [28], combined with our observations can enable one to prove that S_2^2 (or a similar system) cannot prove that BFE requires TC^0 circuits of size $n^{1+\epsilon}$.

7 Inapproximability of MAX-CLIQUE

In this section we use the technique of Srinivasan [30] to show that separating small depth circuits from NP would follow from certain inapproximability results for circuits of nearly-linear size.

For functions $f : \{0,1\}^* \to \mathbb{N}$ and $\alpha : \mathbb{N} \to \mathbb{N}$, *a function* $g : \{0,1\}^* \to \mathbb{N}$ α*-approximates* f if $g(x) \le f(x)$ and $f(x) \le \alpha(|x|)g(x)$ for all $x \in \{0,1\}^*$. MAX-CLIQUE is the following computational problem: given an undirected graph G determine the size of the largest clique in G. For simplicity we assume that G is given by its adjacency matrix and by the size n of G we will understand the number of vertices in G. It is known [17] (see also [21, 22]) that if there is a $n^{1-\epsilon}$-approximation to MAX-CLIQUE computable in ZPP then NP = ZPP. Here ZPP stands for the class of problems solvable on probabilistic machines with zero error in expected polynomial time.

We use the technique of Srinivasan to show the following statement:

Theorem 20 *Let $k > 1$ and let $\epsilon = \epsilon(n)$ be such that $1 > \epsilon(n) = \omega(\log\log n/\log n)$. If* MAX-CLIQUE *is computable by* ACC^0 *circuits of size $O(n^k)$ then a $n^{1-\epsilon(n)}$-approximation to* MAX-CLIQUE *is computable by* ACC^0 *circuits of size $O(n^{1+(k-1)\epsilon(n)})$. Similarly for* TC^0 *and* NC^1 *in place of* ACC^0.

It is interesting to note that the depth of the $O(n^{1+(k-1)\epsilon(n)})$-size circuits does not increase while decreasing $\epsilon(n)$. As stated, the theorem holds only for non-uniform circuits, but a uniform version holds for any function $\epsilon(n)$ that is sufficiently easy to compute.

Proof. Let us assume that we have an ACC^0 circuit family of size $O(n^k)$ that computes MAX-CLIQUE. We will build an ACC^0 circuit family of size $O(n^{1+(k-1)\epsilon})$ computing a $n^{1-\epsilon}$-approximation of MAX-CLIQUE. (For simplicity we assume that $n^{1-\epsilon}$ as well as n^{ϵ} are both integral.) The computation of the approximation proceeds as follows: we partition the vertices of the graph G into $n^{1-\epsilon}$ parts $V_1, \dots, V_{n^{1-\epsilon}}$ of size between $n^{\epsilon} - 1$ and n^{ϵ}. For $i = 1, \dots, n^{1-\epsilon}$ we compute in parallel MAX-CLIQUE of G restricted to V_i. Then we output the largest of these partial results. The correctness of the algorithm follows from the simple observation that if G contains a clique of size $f(G)$ then for some i, V_i contains at least $f(G)/n^{1-\epsilon}$ vertices of that clique and hence MAX-CLIQUE of G restricted to V_i is at least $f(G)/n^{1-\epsilon}$.

The size of a circuit carrying out the computation can be bounded as follows. We use $n^{1-\epsilon}$ circuits of size $O(n^{\epsilon k})$ to compute the value of the $n^{1-\epsilon}$ MAX-CLIQUE subproblems. This already accounts for size $O(n^{1+(k-1)\epsilon})$ and dominates the size of the rest of the construction detailed below. In addition, we need some circuitry to find the largest value among the $n^{1-\epsilon}$ partial results with values from $\{1, \dots, n^{\epsilon}\}$. This can be done using $O(n \log n)$ wires as follows. For each $j \in \{1, \dots, n^{\epsilon}\}$ decide whether the value j appears among the results. This requires $n^{\epsilon} \cdot n^{1-\epsilon}$ comparisons of numbers having $\epsilon \log n$ bits, and thus can be done using $O(n \log n)$ wires. Hence we obtain an indicator vector showing which values from $\{1, \dots, n^{\epsilon}\}$ appear among the partial results. To obtain the resulting value in unary we can use the circuit of Chandra, Fortune, Lipton [11] for computing suffix-OR of the indicator vector using $O(n^{\epsilon})$ wires (if implemented by either ACC^0, TC^0 or NC^1 circuits.) Converting this unary value to binary costs at most $O(n^{\epsilon} \log n)$ additional wires. Hence the size of the circuits is bounded by $O(n^{1+(k-1)\epsilon})$. $\square$

The technique from the previous proof can be also used to establish the following claim.

Theorem 21 *Let $0 < \epsilon < 1$ and $k < 1/\epsilon$ be constants. If there is a polynomial time algorithm that solves* MAX-CLIQUE$_n$ *using an oracle for $m^{1-\epsilon}$-approximation of* MAX-CLIQUE$_m$*, where $m \le n^k$, then* MAX-CLIQUE$_n$ *is downward self-reducible to* MAX-CLIQUE$_{n^{\epsilon k}}$.

Proof. In the proof of Theorem 20 we have seen how to compute a $m^{1-\epsilon}$-approximation of MAX-CLIQUE$_m$ by asking queries to MAX-CLIQUE$_{m^{\epsilon}}$. If there is a polynomial time algorithm that solves MAX-CLIQUE$_n$ using an oracle for $m^{1-\epsilon}$-approximation of MAX-CLIQUE$_m$ where $m \le n^k$, then we can combine it with the above reduction to obtain the desired self-reduction. $\square$

This gives rise to what is perhaps the first example of a lower bound, showing that there is no "quick" reduction between two natural NP-optimization problems. For many natural NP-complete problems A and B, very efficient reductions between A and B are known. (For example, for any problem $A \in$ NTIME$(n \log^{O(1)} n)$, there is a many-one reduction from A to SAT that is computable in time $O(n \log^{O(1)} n)$ [12].) It is easy to show that if $B \notin$ NTIME(n^k), then any reduction from B to SAT requires time n^k – but this does not provide any useful lower bound on the complexity of reducing natural problems to SAT, since no natural NP-complete problem is known to lie outside of NTIME(n). There seems to be no pair of natural NP-complete problems A and B known, where a reduction from A to B is known to require more than linear time (even under the assumption that P $\neq$ NP).

In contrast to this, consider the problem of computing a $n^{1/3}$-approximation to MAX-CLIQUE. Håstad presents a deterministic polynomial-time Turing reduction from MAX-CLIQUE to this approximation problem [17, Theorem 5.3]. (More precisely, Håstad shows that SAT satisfies a condition that implies that a reduction of Bellare, Goldreich, and Sudan correctly reduces SAT to this approximation problem [8]. The polynomial-time Turing reduction from MAX-CLIQUE follows from the trivial observation that MAX-CLIQUE is computable in P$^{\text{NP}}$.) How long must the queries in this reduction be? Assuming that P $\neq$ NP, Theorems 21 and 18 tell us that the queries in this reduction must ask about graphs with at least $n^{1.5}$ vertices.

Corollary 22
- *If* P $\neq$ NP*, any deterministic polynomial-time Turing reduction from* MAX-CLIQUE *to the problem of computing a $n^{1/3}$-approximation to* MAX-CLIQUE *must ask queries of* MAX-CLIQUE$_m$ *for some $m > n^\delta$, if $\delta < 1.5$.*
- *If* P $\neq$ NP*, any deterministic polynomial-time Turing reduction from* MAX-CLIQUE *to the problem of computing a $\sqrt{n}$-approximation to* MAX-CLIQUE *must ask queries of* MAX-CLIQUE$_m$ *for some $m > n^\delta$, if $\delta < 2$.*

Proof. If there is a reduction from MAX-CLIQUE to the problem of computing a $n^{1/3}$-approximation to MAX-CLIQUE that asks queries only to graphs of size n^δ for some $\delta < 1.5$, then by Theorem 21, MAX-CLIQUE$_n$ is downward self-reducible to MAX-CLIQUE$_{n^{\delta \cdot 2/3}}$. By Theorem 18, this implies that MAX-CLIQUE is computable in polynomial time, and hence NP $=$ P.

Part two follows via an essentially identical argument: If there is a reduction from MAX-CLIQUE to the problem of computing a $\sqrt{n}$-approximation to MAX-CLIQUE that asks queries only to graphs of size n^δ for some $\delta < 2$, then by Theorem 21, MAX-CLIQUE$_n$ is downward self-reducible to MAX-CLIQUE$_{n^{\delta/2}}$. By Theorem 18, this implies that MAX-CLIQUE is computable in polynomial time, and hence NP $=$ P.

Clearly, analogous statements can be proved for n^ϵ-approximation for any value of ϵ; the cases $\epsilon \in \{1/3, 1/2\}$ are likely to be of greatest interest. Note that, unlike the case for $\epsilon = 1/3$, there is currently no known deterministic polynomnial-time reduction from MAX-CLIQUE to the problem of computing a $\sqrt{n}$-approximation to MAX-CLIQUE. Similar claims can be also proved for probabilistic reductions instead of deterministic ones. □

It is worthwhile mentioning that, in some sense, decreasing the size of the query length in Håstad's reduction [17] from MAX-CLIQUE to computing a $n^{1/3}$-approximation to MAX-CLIQUE is a *universal* approach to proving P $=$ NP. If any approach will work, then this approach will.

Corollary 23 P $=$ NP *if and only if there is a $\delta < 1.5$ and a deterministic polynomial-time Turing reduction from* MAX-CLIQUE *to the problem of computing a $n^{1/3}$-approximation to* MAX-CLIQUE *that asks queries of size no greater n^δ.*

Proof. One direction follows from Corollary 22. The other direction follows from the observation that if P $=$ NP then there is a polynomial-time Turing reduction for this problem that asks queries of size $O(1)$ (or asks no queries at all). □

8 Circuit lower bounds

We begin this section by showing that problems with small constant-depth circuits have algorithms that run quickly and have small space bounds. Let TISP($t(n), s(n)$) denote the class of problems that are computable by machines running in time $t(n)$ that use space at most $s(n)$. (This definition is somewhat sensitive to the underlying model of computation. We shall always refer explicitly to either the Turing machine model or the random access machine model, to clarify which class is meant.)

Theorem 24 *If A has Dlogtime-uniform* TC^0*circuits of depth d with $O(n^{1+\epsilon})$ wires then for every $0 < \delta < 1+\epsilon$, $A \in TISP((n^{1+\epsilon} + n^{\delta d}) \log^{O(1)} n, n^{1+\epsilon-\delta} \log^{O(1)} n)$ on random access machines and $A \in TISP((n^{1+\epsilon+\delta d} \log^{O(1)} n, n^{1+\epsilon-\delta} \log^{O(1)} n)$ on Turing machines. (The same claim holds with* "TC^0" *replaced by* "ACC^0" *and* "CC^0"*, etc.)*

Proof. A naïve recursive way to evaluate the circuit in space $O(\log n)$ would require time $O(n^{d(1+\epsilon)} \log n)$. Since we can use more space we will use it to remember the computed values of gates that have fan-in larger than n^δ. The faster algorithm then will also recursively evaluate the circuit but whenever it computes the value of a gate with fan-in larger than n^δ it records the value so such a gate will be evaluated at most once. On a random access machine we will store the values in a binary search tree, on a Turing machine we will store them in a simple list. Since there are at most $O(n^{1+\epsilon}/n^\delta)$ gates with fan-in larger than n^δ we will need space only $O(n^{1+\epsilon-\delta} \log^{O(1)} n)$. Finding the value of a gate and whether it has already been computed will take $O(\log^{O(1)} n)$ time on a random access machine and $O(n^{1+\epsilon-\delta} \log^{O(1)} n)$ on a Turing machine. To bound the total time needed to evaluate the circuit notice that we will have to recursively evaluate a tree of fan-in at most n^δ and depth d. To traverse the tree we will need to make $n^{\delta d}$ visits to the nodes. Beside that we will have to evaluate the gates with large fan-in. Since there are at most $O(n^{1+\epsilon})$ wires leading into them these gates will additionally cost at most $O(n^{1+\epsilon})$ node visits. This yields the claimed time bound. □

We need to make use of known time-space tradeoffs for SAT. The following theorem is a special case of Theorem 1.3 in the excellent survey article by van Melkebeek [31]:

Theorem 25 *For every real c such that $1 < c < 5/3$, there exists a positive real e such that SAT cannot be solved by both*

1. *a Π_1 machine with random access that runs in time n^c and*
2. *a deterministic random-access machine that runs in time $n^{1.5}$ and space n^e.*

Moreover, the constant e approaches 1 from below when c approaches 1 from above.

Theorem 26 *For every d there is a constant $\epsilon > 0$ such that SAT does not have Dlogtime-uniform depth d* TC^0 *circuits with fewer than $n^{1+\epsilon}$ wires.*

Proof. Assume that the claim fails for some depth d; thus for every $\epsilon > 0$, SAT has Dlogtime-uniform depth d TC^0 circuits with fewer than $n^{1+\epsilon}$ wires.

By Theorem 24, this implies that for all small ϵ and δ, SAT is in TISP($n^{1+\epsilon} + n^{d\delta}, n^{1+\epsilon-\delta}$). In particular, this is true if we pick $\delta = 2\epsilon$; hence we conclude that for all small enough $\epsilon > 0$, SAT is in TISP($n^{1+\epsilon}, n^{1-\epsilon}$). Since this is true for all ϵ, we have in particular that SAT is in DTIME(n^c) for all $c > 1$.

Pick $\epsilon < \frac{1}{2}$. We thus have SAT is in TISP($n^{1.5}, n^{1-\epsilon}$).

By Theorem 25, if we let c approach 1 from above, the value of e (in Theorem 25) approaches 1 from below. Thus there is some value of $c > 1$ for which $e > 1 - \epsilon$ (in the statement of Theorem 25). Fix these values of c and e. Summarizing, we now have that SAT is in TISP($n^{1.5}, n^e$).

At this point, by Theorem 25, we know that SAT is not in both Π_1 Time(n^c) and TISP($n^{1.5}, n^e$). But we have already observed (three paragraphs ago) that SAT is in DTIME(n^c) and thus it is in Π_1 Time(n^c). Thus we must conclude that SAT is not in TISP($n^{1.5}, n^e$). But this contradicts the conclusion of the preceding paragraph. □

9 Conclusions and open problems

The most important and interesting question raised by this work, is the question of whether it can ultimately lead to separations of complexity classes. (This topic is also discussed in a recent survey [3].) However, a number of other questions naturally arise. We close by listing two such questions.

- Are there sets complete for every level of the NC hierarchy that are downward self-reducible to instances of size n^ϵ? Or is there some fundamental reason why we were unable to find a downward self-reduction of this sort for any problem that is complete for NL or L?
- If NP $=$ TC^0, does SAT have TC^0 circuits of quadratic size? If NEXP $\subseteq$ non-uniform CC^0[6], does the standard complete set for NEXP have CC^0[6] circuits of quadratic size? (Even if arguments based on downward self-reducibility fail for problems outside of NC, perhaps there is another approach that leads to the same conclusion.)

Acknowledgments

We thank David Mix Barrington, Scott Diehl, Lance Fortnow, Stephan Holzer, Alexander Razborov, Mike Saks, Mario Szegedy, Denis Thérien, and Fengming Wang for helpful discussions. We thank Ryan Williams for calling our attention to the work of Srinivasan [30]. We thank the program committee (and their referees) for helpful suggestions.

References

[1] AJTAI, M. A non-linear time lower bound for Boolean branching programs. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)* (1999), pp. 60–70.

[2] ALLENDER, E. Arithmetic circuits and counting complexity classes. In *Complexity of Computations and Proofs*, J. Krajíček, Ed., vol. 13 of *Quaderni di Matematica*. Seconda Università di Napoli, 2004, pp. 33–72.

[3] ALLENDER, E. Cracks in the defenses: Scouting out approaaches on circuit lower bounds. In *Computer Science – Theory and Applications (CSR 2008)* (2008), vol. 5010 of *Lecture Notes in Computer Science*, pp. 3–10.

[4] ALLENDER, E., AND OGIHARA, M. Relationships among PL, #L, and the determinant. *RAIRO - Theoretical Informatics and Applications 30*, 1 (1996), 1–21.

[5] BARRINGTON, D. A. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences 38*, 1 (Feb. 1989), 150–164.

[6] BARRINGTON, D. M., IMMERMAN, N., AND STRAUBING, H. On uniformity within NC^1. *Journal of Computer and System Sciences 41*, 3 (Dec. 1990), 274–306.

[7] BEAME, P., SAKS, M., SUN, X., AND VEE, E. Super-linear time-space tradeoff lower bounds for randomized computation. *Journal of the ACM 50* (2003), 154–195.

[8] BELLARE, M., GOLDREICH, O., AND SUDAN, M. Free bits, pcps, and nonapproximability-towards tight results. *SIAM J. Comput. 27*, 3 (1998), 804–915.

[9] BUSS, S., COOK, S., GUPTA, A., AND RAMACHANDRAN, V. An optimal parallel algorithm for formula evaluation. *SIAM Journal on Computing 21* (1992), 755–780.

[10] CAUSSINUS, H., MCKENZIE, P., THÉRIEN, D., AND VOLLMER, H. Nondeterministic NC^1 computation. *Journal of Computer and System Sciences 57* (1998), 200–212.

[11] CHANDRA, A. K., FORTUNE, S., AND LIPTON, R. J. Lower bounds for constant depth circuits for prefix problems. In *Proc. of the 10th Intl. Colloquium on Automata, Languages and Programming* (1983), vol. 154 of *Lecture Notes in Computer Science*, pp. 109–117.

[12] COOK, S. A. Short propositional formulas represent nondeterministic computations. *Inf. Process. Lett. 26*, 5 (1988), 269–270.

[13] FORTNOW, L. Time-space tradeoffs for satisfiability. *Journal of Computer and System Sciences 60* (2000), 336–353.

[14] FURST, M., SAXE, J. B., AND SIPSER, M. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory 17* (1984), 13–27.

[15] GOLDWASSER, S., GUTFREUND, D., HEALY, A., KAUFMAN, T., AND ROTHBLUM, G. N. Verifying and decoding in constant depth. In *Proc. ACM Symp. on Theory of Computing (STOC)* (2007), pp. 440–449.

[16] HÅSTAD, J. The shrinkage exponent of de Morgan formulas is 2. *SIAM J. Comput. 27*, 1 (1998), 48–64.

[17] HÅSTAD, J. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica 182* (1999), 105–142.

[18] HÅSTAD, J. *Computational Limitations of Small Depth Circuits*. MIT Press, Cambridge, 1988.

[19] HESSE, W., ALLENDER, E., AND BARRINGTON, D. A. M. Uniform constant-depth threshold circuits for division and iterated multiplication. *Journal of Computer and System Sciences 65* (2002), 695–716.

[20] IMPAGLIAZZO, R., PATURI, R., AND SAKS, M. E. Size-depth tradeoffs for threshold circuits. *SIAM J. Comput. 26* (1997), 693–707.

[21] KHOT, S. Improved inaproximability results for maxclique, chromatic number and approximate graph coloring. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)* (2001), pp. 600–609.

[22] KHOT, S., AND PONNUSWAMI, A. K. Better inapproximability results for maxclique, chromatic number and min-3lin-deletion. In *Proc. of the 33rd Intl. Colloquium on Automata, Languages and Programming* (2006), vol. 4051 of *Lecture Notes in Computer Science*, pp. 226–237.

[23] KOUCKÝ, M. Circuit complexity of regular languages. *Theory of Computing Systems* (2008). To appear.

[24] KOUCKÝ, M., PUDLÁK, P., AND THÉRIEN, D. Bounded-depth circuits: Separating wires from gates. In *Proc. ACM Symp. on Theory of Computing (STOC)* (2005), pp. 257–265.

[25] NAOR, M., AND REINGOLD, O. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM 51*, 2 (2004), 231–262.

[26] RAZBOROV, A., AND RUDICH, S. Natural proofs. *J. Comput. Syst. Sci. 55* (1997), 24–35.

[27] RAZBOROV, A. A. Bounded arithmetic and lower bounds. In *Feasible Mathematics II,*, P. Clote and J. Remmel, Eds., vol. 13 of *Progress in Computer Science and Applied Logic*. Birkhäuser, 1995, pp. 344–386.

[28] RAZBOROV, A. A. Unprovability of lower bounds on circuit size in certain fragments of bounded arithmetic. *Izvestiya Math. 59* (1995), 205–227.

[29] SMOLENSKY, R. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proc. ACM Symp. on Theory of Computing (STOC)* (1987), pp. 77–82.

[30] SRINIVASAN, A. On the approximability of clique and related maximization problems. *J. Comput. Syst. Sci. 67*, 3 (2003), 633–651.

[31] VAN MELKEBEEK, D. A survey of lower bounds for satisfiability and related problems. *Foundations and Trends in Theoretical Computer Science 2* (2007), 197–303.

[32] VOLLMER, H. *Introduction to Circuit Complexity*. Springer, 1999.

[33] YAO, A. C. C. Separating the polynomial-time hierarchy by oracles. In *Proceedings 26th Foundations of Computer Science* (1985), IEEE Computer Society Press, pp. 1–10.