Oracles Are Subtle But Not Malicious

Scott Aaronson*
University of Waterloo

Abstract

Theoretical computer scientists have been debating the role of oracles since the 1970's. This paper illustrates both that oracles can give us nontrivial insights about the barrier problems in circuit complexity, and that they need not prevent us from trying to solve those problems.

First, we give an oracle relative to which PP *has linear-sized circuits, by proving a new lower bound for perceptrons and low-degree threshold polynomials. This oracle settles a longstanding open question, and generalizes earlier results due to Beigel and to Buhrman, Fortnow, and Thierauf. More importantly, it implies the first provably nonrelativizing separation of "traditional" complexity classes, as opposed to interactive proof classes such as* MIP *and* $\mathsf{MA}_{\mathsf{EXP}}$. *For Vinodchandran showed, by a nonrelativizing argument, that* PP *does not have circuits of size* n^k *for any fixed* k. *We present an alternative proof of this fact, which shows that* PP *does not even have quantum circuits of size* n^k *with quantum advice. To our knowledge, this is the first nontrivial lower bound on quantum circuit size.*

Second, we study a beautiful algorithm of Bshouty et al. for learning Boolean circuits in $\mathsf{ZPP}^{\mathsf{NP}}$. *We show that the* NP *queries in this algorithm cannot be parallelized by any relativizing technique, by giving an oracle relative to which* $\mathsf{ZPP}_{||}^{\mathsf{NP}}$ *and even* $\mathsf{BPP}_{||}^{\mathsf{NP}}$ *have linear-size circuits. On the other hand, we also show that the* NP *queries could be parallelized if* $\mathsf{P} = \mathsf{NP}$. *Thus, classes such as* $\mathsf{ZPP}_{||}^{\mathsf{NP}}$ *inhabit a "twilight zone," where we need to distinguish between relativizing and black-box techniques. Our results on this subject have implications for computational learning theory as well as for the circuit minimization problem.*

1. Introduction

It is often lamented that, half a century after Shannon's insight [36] that almost all Boolean functions require exponential-size circuits, there is still no explicit function for which we can prove even a superlinear lower bound. Yet whether this lament is justified depends on what we mean by "explicit." For in 1982, Kannan [20] did show that for every constant k, there exists a language in Σ_2^p (the second level of the polynomial hierarchy) that does not have circuits of size n^k. His proof used the oldest trick in the book: diagonalization, defined broadly as any method for simulating all machines in one class by a single machine in another. In some sense, diagonalization is still the only method we know that zeroes in on a "non-natural" property of the function being lower-bounded (loosely speaking, a property that does not hold of a random function), and thereby escapes the jaws of Razborov and Rudich [32].

But can we generalize Kannan's theorem to other complexity classes? A decade ago, Bshouty et al. [9] discovered an algorithm to learn Boolean circuits in $\mathsf{ZPP}^{\mathsf{NP}}$ (that is, probabilistic polynomial time with NP oracle). As shown by Köbler and Watanabe [23], the existence of this algorithm implies that $\mathsf{ZPP}^{\mathsf{NP}}$ itself cannot have circuits of size n^k for any k.[1]

So our task as lowerboundsmen and lowerboundswomen seems straightforward: namely, to find increasingly powerful algorithms for learning Boolean circuits, which can then be turned around to yield increasingly powerful circuit lower bounds. But when we try to do this, we quickly run into the brick wall of relativization. Just as Baker, Gill, and Solovay [7] gave a relativized world where $\mathsf{P} = \mathsf{NP}$, so Wilson [46] gave relativized worlds where NP and P^{NP} have linear-size circuits. Since the results of Kannan [20] and Bshouty et al. [9] relativize, this suggests that new techniques will be needed to make further progress.

Yet attitudes toward relativization vary greatly within our community. Some computer scientists ridicule oracle results as elaborate formalizations of the obvious—apparently believing that (1) there exist relativized worlds where just

*Email: scott@scottaaronson.com. Most of this work was done while the author was a postdoc at the Institute for Advanced Study in Princeton, supported by an NSF grant.

[1] For Bshouty et al.'s algorithm implies the following improvement to the celebrated Karp-Lipton theorem [21]: if $\mathsf{NP} \subset \mathsf{P/poly}$ then PH collapses to $\mathsf{ZPP}^{\mathsf{NP}}$. There are then two cases: if $\mathsf{NP} \not\subset \mathsf{P/poly}$, then certainly $\mathsf{ZPP}^{\mathsf{NP}} \not\subset \mathsf{P/poly}$ as well and we are done. On the other hand, if $\mathsf{NP} \subset \mathsf{P/poly}$, then $\mathsf{ZPP}^{\mathsf{NP}} = \mathsf{PH}$, but we already know from Kannan's theorem that PH does not have circuits of size n^k. Indeed, we can repeat this argument for the class S_2^p, which Cai [12] showed is contained in $\mathsf{ZPP}^{\mathsf{NP}}$.

about anything is true, (2) the creation of such worlds is a routine exercise, (3) the only conjectures ruled out by oracle results are trivially false ones, which no serious researcher would waste time trying to prove, and (4) nonrelativizing results such as $\mathsf{IP} = \mathsf{PSPACE}$ [35] render oracles irrelevant anyway. At the other extreme, some computer scientists see oracle results not as a spur to create nonrelativizing techniques or as a guide to where such techniques might be needed, but as an excuse to abandon hope.

This paper will offer new counterexamples to both of these views, in the context of circuit lower bounds. We focus on two related topics: first, the classical and quantum circuit complexity of PP; and second, the learnability of Boolean circuits using parallel NP queries.

1.1. On PP and Quantum Circuits

In Section 2, we give an oracle relative to which PP has linear-size circuits. Here PP is the class of languages accepted by a nondeterministic polynomial-time Turing machine that accepts if and only if the majority of its paths do. Our construction also yields an oracle relative to which PEXP (the exponential-time version of PP) has polynomial-size circuits, and indeed $\mathsf{P}^{\mathsf{NP}} = \oplus\mathsf{P} = \mathsf{PEXP}$. This settles several questions that were open for years,[2] and subsumes at least four previous results: that of Beigel [8] giving an oracle relative to which $\mathsf{P}^{\mathsf{NP}} \not\subset \mathsf{PP}$ (since clearly $\mathsf{P}^{\mathsf{NP}} = \mathsf{PEXP}$ implies $\mathsf{P}^{\mathsf{NP}} \not\subset \mathsf{PP}$); that of Aspnes et al. [5] giving an oracle relative to which $\mathsf{PP} \neq \mathsf{PSPACE}$ (since PSPACE does not have linear-size circuits relative to any oracle);[3] that of Buhrman, Fortnow, and Thierauf [11] giving an oracle relative to which $\mathsf{MA}_{\mathsf{EXP}} \subset \mathsf{P}/\mathsf{poly}$; and that of Buhrman et al. [10] giving an oracle relative to which $\mathsf{P}^{\mathsf{NP}} = \mathsf{NEXP}$.

Note that our result is nearly optimal, since Toda's theorem [40] yields a relativizing proof that P^{PP} and even $\mathsf{BP} \cdot \mathsf{PP}$ do not have circuits of any fixed polynomial size.

Our proof first represents each PP machine by a low-degree multilinear polynomial, whose variables are the bits of the oracle string. It then combines these polynomials into a single polynomial called Q. The key fact is that, if there are no variables left "unmonitored" by the component polynomials, then we can modify the oracle in a way that increases Q. Since Q can only increase a finite number of times, it follows that we will eventually win our "war of attrition" against the polynomials, at which point we can simply write down what each machine does in an unmonitored part of the oracle string. The main novelty of the proof lies in how we combine the polynomials to create Q.

[2] Lance Fortnow, personal communication.

[3] Admittedly, our result does not imply that $\mathsf{PP} \neq \mathsf{PSPACE}$ relative to a *random* oracle with probability 1.

We can state our result alternatively in terms of *perceptrons* [28], also known as threshold-of-AND circuits or polynomial threshold functions. Call a perceptron "small" if it has size $2^{N^{o(1)}}$, order $N^{o(1)}$, and weights in $\{-1, 1\}$. Also, given an N-bit string $x_1 \ldots x_N$, recall that the ODDMAXBIT problem is to decide whether the maximum i such that $x_i = 1$ is even or odd, promised that such an i exists. Then Beigel [8] showed that no small perceptron can solve ODDMAXBIT. What we show is a strong generalization of Beigel's theorem: for any $k = N^{o(1)}$ small perceptrons, there exists a "problem set" consisting of k ODDMAXBIT instances, such that for every j, the j^{th} perceptron will get the j^{th} problem wrong even if it can examine the whole problem set. Previously this had been open even for $k = 2$.

But the real motivation for our result is that in the unrelativized world, PP is known *not* to have linear-size circuits. Indeed, Vinodchandran [45] showed that for every k, there exists a language in PP that does not have circuits of size n^k. Putting our result together with Vinodchandran's, we obtain what appears to be the first nonrelativizing separation that does not involve artificial classes or classes defined using interactive proofs. There have been nonrelativizing separations in the past, but most of them have followed easily from the collapse of interactive proof classes:[4] for example, $\mathsf{NP} \neq \mathsf{MIP}$ from $\mathsf{MIP} = \mathsf{NEXP}$ [6], and $\mathsf{IP} \not\subset \mathsf{SIZE}\left(n^k\right)$ from $\mathsf{IP} = \mathsf{PSPACE}$ [35]. The one exception was the result of Buhrman, Fortnow, and Thierauf [11] that $\mathsf{MA}_{\mathsf{EXP}} \not\subset \mathsf{P}/\mathsf{poly}$, where $\mathsf{MA}_{\mathsf{EXP}}$ is the exponential-time version of MA. However, the class $\mathsf{MA}_{\mathsf{EXP}}$ exists for the specific purpose of not being contained in P/poly, and the resulting separation does not scale down below NEXP, to show (for example) that MA does not have linear-size circuits.

The actual lower bound of Vinodchandran [45] follows easily from three well-known results: the LFKN interactive protocol for the permanent [26], Toda's theorem [40], and Kannan's theorem [20].[5] In Section 3, we present an alternative, more self-contained proof, which does not go through Toda's theorem. As a bonus, our proof also shows that PP does not have *quantum* circuits of size n^k for any k. Indeed, this remains true even if the quantum circuits are given "quantum advice states" on n^k qubits. One part of our proof is a "quantum Karp-Lipton theorem," which

[4] Note that we do not count separations that depend on a specific machine model, such as the result of Paul et al. [31] that $\mathsf{DTIME}(n) \neq \mathsf{NTIME}(n)$ for multitape Turing machines.

[5] Suppose by contradiction that PP has circuits of size n^k. Then $\mathsf{P}^{\#\mathsf{P}} \subset \mathsf{P}/\mathsf{poly}$, and therefore $\mathsf{MA} = \mathsf{P}^{\#\mathsf{P}}$ by a result of LFKN [26] (this is the only part of the proof that fails to relativize). Now $\mathsf{MA} \subseteq \Sigma_2^p \subseteq \mathsf{P}^{\#\mathsf{P}}$ by Toda's theorem [40], and $\mathsf{MA} \subseteq \mathsf{PP} \subseteq \mathsf{P}^{\#\mathsf{P}}$ by an observation of Vereshchagin [44]. Therefore $\Sigma_2^p = \mathsf{PP}$ as well. But we already know from Kannan's theorem [20] that Σ_2^p does not have circuits of size n^k.

states that if PP has polynomial-size quantum circuits, then the "counting hierarchy" (consisting of PP, $\mathsf{PP}^{\mathsf{PP}}$, $\mathsf{PP}^{\mathsf{PP}^{\mathsf{PP}}}$, and so on) collapses to QMA, the quantum analogue of NP. By analogy to the classical nonrelativizing separation of Buhrman, Fortnow, and Thierauf [11], we also show that $\mathsf{QMA}_{\mathsf{EXP}}$, the exponential-time version of QMA, is not contained in $\mathsf{BQP/qpoly}$. Indeed, $\mathsf{QMA}_{\mathsf{EXP}}$ requires quantum circuits of at least "half-exponential" size, meaning size $f(n)$ where $f(f(n))$ grows exponentially.[6] So far as we know, the only previous lower bounds for arbitrary quantum circuits were due to Nishimura and Yamakami [30], who showed (among other things) that $\mathsf{EESPACE} \not\subset \mathsf{BQP/qpoly}$.

1.2. On Parallel NP Queries and Black-Box Learning

In a second part of the paper, we study the algorithm of Bshouty et al. [9] for learning Boolean circuits. Given a Boolean function f that is promised to have a polynomial-size circuit, this algorithm *finds* such a circuit in the class $\mathsf{ZPP}^{\mathsf{NP}^f}$: that is, zero-error probabilistic polynomial time with NP oracle with oracle for f. One of the most basic questions about this algorithm is whether the NP queries can be parallelized. For if so, then we immediately obtain a new circuit lower bound: namely that $\mathsf{ZPP}_{||}^{\mathsf{NP}}$ (that is, ZPP with parallel NP queries) does not have circuits of size n^k for any k.[7] Conceptually, this would not be so far from showing that NP itself does not have circuits of size n^k.[8]

Let $\mathcal{C}$ be the set of circuits of size n^k. In Bshouty et al.'s algorithm, we repeatedly ask the NP oracle to find us an input x_t such that, among the circuits in $\mathcal{C}$ that succeed on all previous inputs $x_1, \ldots, x_{t-1}$, at least a $1/3$ fraction fail on x_t. Since each such input reduces the number of circuits "still in the running" by at least a constant factor, this process can continue for at most $\log|\mathcal{C}|$ steps. Furthermore, when it ends, by assumption we have a set $\mathcal{C}^*$ of circuits such that for all inputs x, a uniform random circuit drawn from $\mathcal{C}^*$ will succeed on x with probability at least $2/3$. So now all we have to do is sample a polynomial number of circuits from $\mathcal{C}^*$, then generate a new circuit that outputs the majority answer among the sampled circuits. The technical part is to express the concepts "at least a $1/3$ fraction" and "a uniform random sample" in NP. For that, Bshouty et al. use pairwise-independent hash functions.

When we examine Bshouty et al.'s algorithm, it is far from obvious that adaptive NP queries are necessary. For why can't we simply ask the following question in parallel, for all $T \leq \log|\mathcal{C}|$?

> "Do there exist inputs $x_1, \ldots, x_T$, such that at least a $1/3$ fraction of circuits in $\mathcal{C}$ fail on x_1, and among the circuits that succeed on x_1, at least a $1/3$ fraction fail on x_2, and among the circuits that succeed on x_1 and x_2, at least a $1/3$ fraction fail on $x_3, \ldots$ and so on up to x_T?"

By making clever use of hashing and approximate counting, perhaps we could control the number of circuits that succeed on $x_1, \ldots, x_t$ for all $t \leq T$. In that case, by finding the largest T such that the above question returns a positive answer, and then applying the Valiant-Vazirani reduction [43] and other standard techniques, we would achieve the desired parallelization of Bshouty et al.'s algorithm. Indeed, when we began studying the topic, it seemed entirely likely to us that this was possible.

Nevertheless, in Section 4 we give an oracle relative to which $\mathsf{ZPP}_{||}^{\mathsf{NP}}$ and even $\mathsf{BPP}_{||}^{\mathsf{NP}}$ have linear-size circuits. The overall strategy of our oracle construction is the same as for PP, but the details are different. The existence of this oracle means that any parallelization of Bshouty et al.'s algorithm would need to use nonrelativizing techniques.[9]

[6] See Miltersen, Vinodchandran, and Watanabe [27] for a discussion of this concept.

[7] This follows from an argument similar to that used by Köbler and Watanabe [23] to show that $\mathsf{ZPP}^{\mathsf{NP}}$ does not have circuits of size n^k. In particular, suppose we could learn a circuit for f in $\mathsf{ZPP}_{||}^{\mathsf{NP}^f}$. Then there are two cases: if $\mathsf{NP} \not\subset \mathsf{P/poly}$, then certainly $\mathsf{ZPP}_{||}^{\mathsf{NP}} \not\subset \mathsf{P/poly}$ and we are done. On the other hand, if $\mathsf{NP} \subset \mathsf{P/poly}$, then a $\mathsf{ZPP}_{||}^{\mathsf{NP}}$ machine could learn a polynomial-size circuit for SAT, and then use that circuit to decide any language in PH. So again, $\mathsf{ZPP}_{||}^{\mathsf{NP}}$ would not have circuits of size n^k.

[8] For as observed by Shaltiel and Umans [34] and Fortnow and Klivans [15] among others, there is an intimate connection between the classes $\mathsf{P}_{||}^{\mathsf{NP}}$ and $\mathsf{NP/log}$. Furthermore, any circuit lower bound for $\mathsf{NP/log}$ implies the same lower bound for NP, since we can tack the advice onto the input.

[9] As a side note, researchers in learning theory often allow "equivalence queries" to the target function f. Given a circuit C, an equivalence query returns 'YES' if C computes f, and otherwise returns any $x \in \{0,1\}^n$ such that $C(x) \neq f(x)$. When we said that any parallelization of Bshouty et al.'s algorithm would need to be nonrelativizing, it might be objected that we ignored the possibility of parallel equivalence queries. However, we can *simulate* an equivalence query to f in $\mathsf{ZPP}_{||}^{\mathsf{NP}^f}$, by using the following simple trick. First we choose pairwise-independent hash functions $h_1, \ldots, h_n : \{0,1\}^n \rightarrow \{0,1\}$. Next we make the following queries in parallel to the NP^f oracle:

(1) For all $t \in \{0, \ldots, n\}$: "Does there exist an $x \in \{0,1\}^n$ such that $C(x) \neq f(x)$ and $h_1(x) = \cdots = h_t(x) = 0$?"

(2) For all $t \in \{0, \ldots, n\}$: "Do there exist distinct x, x' satisfying the above conditions?"

(3) For all $t \in \{0, \ldots, n\}$ and $i \in \{0, \ldots, n\}$: "Does there exist an $x = x_1 \ldots x_n$ satisfying the above conditions such that $x_i = 1$?"

Provided there exists a t for which query (1) returns 'YES' and query (2) returns 'NO', we can then read a particular x such that $C(x) \neq f(x)$ off the answers to query (3). Finally, by repeating all of this several times in parallel, we can amplify the probability of success. Note that, if there exists an x such that $C(x) \neq f(x)$ but the algorithm fails to find such an x, then the algorithm knows this and can output 'FAILURE.' Hence, this is indeed a ZPP algorithm.

Yet even here, the situation is subtler than one might imagine. To explain why, we need to distinguish carefully between relativizing and black-box algorithms. An algorithm for learning Boolean circuits is *relativizing* if, when given access to an oracle A, the algorithm can learn circuits that are also given access to A. But a nonrelativizing algorithm can still be *black-box*, in the sense that it learns about the target function f only by querying it, and does not exploit any succinct description of f (for example, that $f(x) = 1$ if and only if x encodes a satisfiable Boolean formula). Bshouty et al.'s algorithm is both relativizing *and* black-box. What our oracle construction shows is that no relativizing algorithm can learn Boolean circuits in $\mathsf{BPP}_{||}^{\mathsf{NP}}$. But what about a nonrelativizing yet still black-box algorithm?

Surprisingly, we show in Section 5 that if $\mathsf{P} = \mathsf{NP}$, then there *is* a black-box algorithm to learn Boolean circuits even in $\mathsf{P}_{||}^{\mathsf{NP}}$ (as well as in $\mathsf{NP}/\log$). Despite the outlandishness of the premise, this theorem is not trivial, and requires many of the same techniques originally used by Bshouty et al. [9]. One way to interpret the theorem is that we cannot show the *impossibility* of black-box learning in $\mathsf{P}_{||}^{\mathsf{NP}}$, without also showing that $\mathsf{P} \neq \mathsf{NP}$. By contrast, it is easy to show that black-box learning is impossible in NP, regardless of what computational assumptions we make.[10]

These results provide a new perspective on one of the oldest problems in computer science, the *circuit minimization problem*: given a Boolean circuit C, does there exist an equivalent circuit of size at most s? Certainly this problem is NP-hard and in Σ_2^p. Also, by using Bshouty et al.'s algorithm, we can find a circuit whose size is within an $O(n/\log n)$ factor of minimal in $\mathsf{ZPP}^{\mathsf{NP}}$. Yet after fifty years of research, almost nothing else is known about the complexity of this problem. For example, is it Σ_2^p-complete? Can we approximate the minimum circuit size in $\mathsf{ZPP}_{||}^{\mathsf{NP}}$?

What our techniques let us say is the following. First, there exists an oracle A such that minimizing circuits with oracle access to A is not even approximable in $\mathsf{BPP}_{||}^{\mathsf{NP}^A}$. Indeed, any probabilistic algorithm to distinguish the cases "C is minimal" and "there exists an equivalent circuit for C of size at most s," using $o(s)$ adaptive NP queries, would have to use nonrelativizing techniques. Intuitively, the reason is as follows. Given a circuit C that computes a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$, there might be a "hidden region" of the oracle string A that contains the truth table of f. In that case, even if C is minimal in the unrelativized world, relative to A there might be a much smaller circuit for f—one that simply encodes the location of that hidden region. However, our results will imply that, if the hidden region takes $s \gg \log n$ bits to specify, then any BPP algorithm needs $\Omega(s)$ adaptive NP^A queries (or $2^{\Omega(s)}$ non-adaptive NP^A queries) to decide whether or not it exists. In particular, this problem is not solvable in $\mathsf{BPP}_{||}^{\mathsf{NP}^A}$.

If one wished, one could take our oracle result as evidence that the true complexity of approximate circuit minimization should be P^{NP}, rather than $\mathsf{P}_{||}^{\mathsf{NP}}$. However, the results of Section 5 suggest that it will be difficult to show (for example) that approximate circuit minimization is P^{NP}-hard. For any hardness proof will have to fight a "two-front war"—firstly against algorithms that exploit the internal structure of a circuit C, and secondly against black-box algorithms (that is, algorithms that treat C as an oracle)! The reason is that, if $\mathsf{P} = \mathsf{NP}$, then there *is* a black-box circuit minimization algorithm in $\mathsf{P}_{||}^{\mathsf{NP}}$.

From a learning theory perspective, perhaps what is most interesting about our results is that they show a clear trade-off between two complexities: the complexity of the learner who queries the target function, and the complexity of the resulting computational problem that the learner has to solve. In particular, suppose a learner is given oracle access to a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ with polynomial circuit complexity, and wants to output a circuit C for f. Then if the learner is a $\mathsf{ZPP}^{\mathsf{NP}^f}$ machine, the computational problem of finding C is easy, as shown by Bshouty et al. [9]. If the learner is a $\mathsf{ZPP}_{||}^{\mathsf{NP}^f}$ machine, then the problem is of finding C is probably hard, as indicated by our results. If the learner is an NP^f machine, then there is *no* computational problem whose solution would suffice to find C.

1.3. Outlook

Figure 1 shows the "battle map" for nonrelativizing circuit lower bounds that emerges from this paper. The figure displays not one but two barriers: a "relativization barrier," below which any Karp-Lipton collapse or superlinear circuit size lower bound will need to use nonrelativizing techniques; and a "black-box barrier," below which black-box learning even of unrelativized circuits is provably impossible. At least for the thirteen complexity classes shown in the figure, we now know exactly where to draw these two barriers—something that would have been less than obvious *a priori* (at least to us!).

To switch metaphors, we can think of the barriers as representing "phase transitions" in the behavior of complexity classes. Below the black-box barrier, we cannot learn circuits relative to any oracle A. Between the relativization and black-box barriers, we can learn Boolean circuits relative to *some* oracles A but not others. For example, we can learn relative to a PSPACE oracle, since it collapses P

[10] Note that by "learn," we always mean "learn exactly using membership queries" rather than "PAC-learn." If $\mathsf{P} = \mathsf{NP}$, then *approximate* learning of Boolean circuits can clearly be done even in P. For we simply query the target function f at a polynomial number of locations drawn from the distribution of interest, then find a small circuit that agrees with f on as many of those locations as possible.

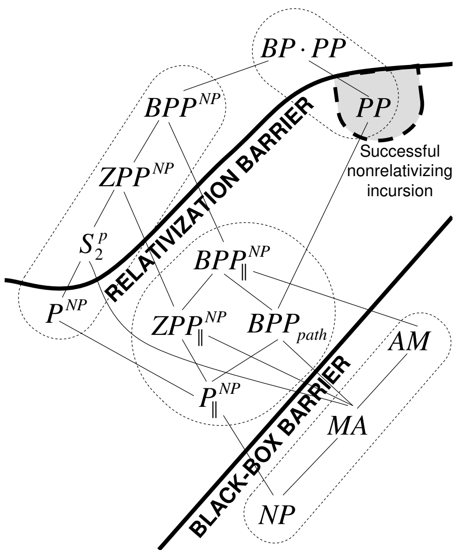

Figure 1. "Battle map" of some complexity classes between NP and BP · PP, in light of this paper's results. Classes that coincide under a plausible derandomization assumption are grouped together with dashed ovals. Below the relativization barrier, we must use nonrelativizing techniques to show any Karp-Lipton collapse or superlinear circuit size lower bound. Below the black-box barrier, black-box learning of Boolean circuits is provably impossible.

and NP, but cannot learn relative to the oracles in this paper, which cause PP and $\mathsf{BPP}_{||}^{\mathsf{NP}}$ to have linear-size circuits. Finally, above the relativization barrier, we can learn Boolean circuits relative to *every* oracle A.[11] As we move upward from the black-box barrier toward the relativization barrier, we can notice "steam bubbles" starting to form, as the assumptions needed for black-box learning shift from implausible ($\mathsf{P} = \mathsf{NP}$), to plausible (the standard derandomization assumptions that collapse P^{NP} with $\mathsf{ZPP}^{\mathsf{NP}}$ and PP with BP · PP), and finally to no assumptions at all.

To switch metaphors again, the oracle results have laid before us a rich and detailed landscape, which a nonrelativizing Lewis-and-Clark expedition might someday visit more fully.

[11] There is one important caveat: in S_2^p, we currently only know how to learn self-reducible functions, such as the characteristic functions of NP-complete problems. For if the circuits from the two competing provers disagree with each other, then we need to know which one to trust.

2. The Oracle for PP

In this section we construct an oracle relative to which PP has linear-size circuits. To do so, we first need a lemma about multilinear polynomials.

Lemma 1 *Let $p : \{0,1\}^N \rightarrow \mathbb{R}$ be a real multilinear polynomial of degree at most $\sqrt{N}/7$, and suppose that $|p(X)| \leq \frac{2}{3}\left|p\left(0^N\right)\right|$ for all $X \in \{0,1\}^N$ with Hamming weight 1. Then there exists an $X \in \{0,1\}^N$ such that $|p(X)| \geq 6\,|p(0^n)|$.*

Lemma 1 follows immediately from the well-known lower bound of Nisan and Szegedy [29] on the approximate degree of the OR function, which in turn built on earlier results of Ehlich and Zeller [13] and Rivlin and Cheney [33].

We can now prove the main result.

Theorem 2 *There exists an oracle relative to which* PP *has linear-size circuits.*

Proof. For simplicity, we first give an oracle that works for a specific value of n, and then generalize to all n simultaneously. Let $M_1, M_2, \ldots$ be an enumeration of $\mathsf{PTIME}\left(n^{\log n}\right)$ machines. Then it suffices to simulate $M_1, \ldots, M_n$, for in that case every M_i will be simulated on all but finitely many n.

The oracle A will consist of 2^{5n} "rows" and $n2^n$ "columns," with each row labeled by a string $r \in \{0,1\}^{5n}$, and each column labeled by a pair $\langle i, x \rangle$ where $i \in \{1, \ldots, n\}$ and $x \in \{0,1\}^n$. Then given a triple $\langle r, i, x \rangle$ as input, A will return the bit $A(r, i, x)$.

We will construct A via an iterative procedure. Initially A is empty (that is, $A(r, i, x) = 0$ for all r, i, x). Let A_t be the state of A after the t^{th} iteration. Also, let $M_{i,x}(A_t)$ be a Boolean function that equals 1 if M_i accepts on input $x \in \{0,1\}^n$ and oracle string A_t, and 0 otherwise. Then to *encode* a row r means to set $A_t(r, i, x) := M_{i,x}(A_{t-1})$ for all i, x. At a high level, our entire procedure will consist of repeating the following two steps, for all $t \geq 1$:

(1) Choose a set of rows $S \subseteq \{0,1\}^{5n}$ of A_{t-1}.

(2) Encode each $r \in S$, and let A_t be the result.

The problem, of course, is that each time we encode a row r, the $M_{i,x}(A_t)$'s might change as a result. So we need to show that, by carefully implementing step (1), we can guarantee that the following condition holds after a finite number of steps t.

($\mathcal{C}$) There exists an r such that $A_t(r, i, x) = M_{i,x}(A_t)$ for all i, x.

If ($\mathcal{C}$) is satisfied, then clearly $M_1, \ldots, M_n$ will have linear-size circuits relative to A_t, since we can just hardwire r into the circuits.

We will use the following fact, which is immediate from the definition of PP. For all i, x, there exists a multilinear polynomial $p_{i,x}(A)$, whose variables are the bits of A, such that:

(i) If $M_{i,x}(A) = 1$ then $p_{i,x}(A) \geq 1$.

(ii) If $M_{i,x}(A) = 0$ then $p_{i,x}(A) \leq -1$.

(iii) $p_{i,x}$ has degree at most $n^{\log n}$.

(iv) $|p_{i,x}(A)| \leq 2^{n^{\log n}}$ for all A.

Now for all integers $0 \leq k \leq n^{\log n}$ and $b \in \{0,1\}$, let

$$q_{i,x,b,k}(A) = 2^{2k-3} + \left(2^k + (-1)^b p_{i,x}(A)\right)^2.$$

Then we will use the following polynomial as a progress measure:

$$Q(A) = \prod_{i,x} \prod_{b \in \{0,1\}} \prod_{k=0}^{n^{\log n}} q_{i,x,b,k}(A).$$

Notice that

$$\deg(Q) \leq n2^n \cdot 2 \cdot \left(n^{\log n} + 1\right) \cdot 2\deg(p_{i,x}) = 2^{n+o(n)}.$$

Since $1/8 \leq q_{i,x,b,k}(A) \leq 5 \cdot 2^{2n^{\log n}}$ for all i, x, b, k, we also have

$$Q(A) \leq \left(5 \cdot 2^{2n^{\log n}}\right)^{n2^n \cdot 2 \cdot \left(n^{\log n}+1\right)} = 2^{2^{n+o(n)}},$$
$$Q(A) \geq \left(\frac{1}{8}\right)^{n2^n \cdot 2 \cdot \left(n^{\log n}+1\right)} = 2^{-2^{n+o(n)}}$$

for all A. The key claim is the following.

At any given iteration, suppose there is no r such that, by encoding r, we can satisfy condition ($\mathcal{C}$). Then there exists a set $S \subseteq \{0,1\}^{5n}$ such that, by encoding each $r \in S$, we can increase $Q(A_t)$ by at least a factor of 2 (that is, ensure that $Q(A_t) \geq 2Q(A_{t-1})$).

The above claim readily implies that ($\mathcal{C}$) can be satisfied after a finite number of steps. For, by what was said previously, $Q(A_t)$ can double at most $2^{n+o(n)}$ times—and once $Q(A_t)$ can no longer double, by assumption we can encode an r that satisfies ($\mathcal{C}$). (As a side note, "running out of rows" is not an issue here, since we can re-encode rows that were encoded in previous iterations.)

We now prove the claim. Call the pair $\langle i, x\rangle$ *sensitive* to row r if encoding r would change the value of $M_{i,x}(A)$. If there exists a row r to which no $\langle i, x\rangle$ is sensitive, then we simply encode that row and are done. Suppose, on the other hand, that for every r there exists an $\langle i, x\rangle$ that is sensitive to r. Then by a counting argument, there exists a single $\langle i, x\rangle$ that is sensitive to at least $2^{5n}/(n2^n) > 2^{3n}$ rows. Fix that $\langle i, x\rangle$, and let $r_1, \ldots, r_{2^{3n}}$ be the first 2^{3n} rows to which $\langle i, x\rangle$ is sensitive. Also, given a binary string $Y = y_1 \ldots y_{2^{3n}}$, let $S(Y)$ be the set of all r_j such that $y_j = 1$, and let $A^{(Y)}$ be the oracle obtained by starting from A and then encoding each $r_j \in S(Y)$.

Set b equal to $M_{i,x}(A)$, and set k equal to the least integer such that $2^k \geq |p_{i,x}(A)|$. Then we will think of $Q(A)$ as the product of two polynomials $q(A)$ and $v(A)$, where $q(A) = q_{i,x,b,k}(A)$, and $v(A) = Q(A)/q(A)$ is the product of all other terms in $Q(A)$. Notice that $q(A) > 0$ and $v(A) > 0$ for all A. Also,

$$\begin{aligned} q(A) &= 2^{2k-3} + \left(2^k + (-1)^b p_{i,x}(A)\right)^2 \\ &\leq 2^{2k-3} + \left(2^k - 2^{k-1}\right)^2 \\ &= \frac{3}{8} \cdot 2^{2k}. \end{aligned}$$

Here the second line follows since $-2^k \leq (-1)^b p_{i,x}(A) \leq -2^{k-1}$. On the other hand, let Y be any 2^{3n}-bit string with Hamming weight 1, so that $A^{(Y)}$ is obtained from A by encoding a single row to which $\langle i, x\rangle$ is sensitive. Then we have $(-1)^b p_{i,x}\left(A^{(Y)}\right) \geq 0$, and therefore

$$\begin{aligned} q\left(A^{(Y)}\right) &= 2^{2k-3} + \left(2^k + (-1)^b p_{i,x}\left(A^{(Y)}\right)\right)^2 \\ &\geq 2^{2k-3} + \left(2^k\right)^2 \\ &= \frac{9}{8} \cdot 2^{2k} \\ &\geq 3q(A). \end{aligned}$$

There are now two cases. The first is that there exists a Y with Hamming weight 1 such that $v\left(A^{(Y)}\right) \geq \frac{2}{3}v(A)$. In this case

$$\begin{aligned} Q\left(A^{(Y)}\right) &= q\left(A^{(Y)}\right) v\left(A^{(Y)}\right) \\ &\geq 3q(A) \cdot \frac{2}{3}v(A) \\ &= 2q(A)v(A) \\ &= 2Q(A). \end{aligned}$$

So we simply set $S = S(Y)$ and are done.

The second case is that $v\left(A^{(Y)}\right) < \frac{2}{3}v(A)$ for all Y with Hamming weight 1. In this case, we can consider v as a real multilinear polynomial in the bits of $Y \in \{0,1\}^{2^{3n}}$, of degree at most $\deg(Q) < \sqrt{2^{3n}}/7$. Then Lemma 1 implies that there exists a $Y \in \{0,1\}^{2^{3n}}$ such that $\left|v\left(A^{(Y)}\right)\right| = v\left(A^{(Y)}\right) \geq 6v(A)$. Furthermore, for all Y we have

$$\frac{q\left(A^{(Y)}\right)}{q(A)} \geq \frac{2^{2k-3}}{\frac{3}{8} \cdot 2^{2k}} = \frac{1}{3}.$$

Hence

$$\begin{aligned}Q\left(A^{(Y)}\right) &= q\left(A^{(Y)}\right) v\left(A^{(Y)}\right)\\ &\geq \frac{1}{3} q\left(A\right) \cdot 6v\left(A\right)\\ &= 2q\left(A\right) v\left(A\right)\\ &= 2Q\left(A\right).\end{aligned}$$

So again we can set $S = S(Y)$. This completes the claim.

All that remains is to handle $\mathsf{PTIME}\left(n^{\log n}\right)$ machines that could query *any* bit of the oracle string, rather than just the bits corresponding to a specific n. The oracle A will now take as input a *list* of strings $R = (r_1, \ldots, r_\ell)$, with $r_\ell \in \{0,1\}^{5 \cdot 2^\ell}$ for all ℓ, in addition to i, x. Call R an *ℓ-secret* if $A(R, i, x) = M_{i,x}(A)$ for all $n \leq 2^\ell$, $i \in \{1, \ldots, n\}$, and $x \in \{0,1\}^n$. Then we will try to satisfy the following.

($\mathcal{C}'$) There exists an infinite list of strings $r_1^*, r_2^*, \ldots,$, such that $R_\ell^* := (r_1^*, \ldots, r_\ell^*)$ is an ℓ-secret for all $\ell \geq 1$.

If ($\mathcal{C}'$) is satisfied, then clearly each M_i can be simulated by linear-size circuits. For all $n \geq i$, simply find the smallest ℓ such that $2^\ell \geq n$, then hardwire R_ℓ^* into the circuit for size n. Since $\ell \leq 2n$, this requires at most $5\left(2^1 + \cdots + 2^\ell\right) \leq 20n$ bits.

To construct an oracle A that satisfies ($\mathcal{C}'$), we iterate over all $\ell \geq 1$. Suppose by induction that $R_{\ell-1}^*$ is an $(\ell-1)$-secret; then we want to ensure that R_ℓ^* is an ℓ-secret for some $r_\ell \in \{0,1\}^{5 \cdot 2^\ell}$. To do so, we use a procedure essentially identical to the one for a specific n. The only difference is this: previously, all we needed was a row $r \in \{0,1\}^{5n}$ such that no $\langle i, x\rangle$ pair was sensitive to a *particular* change to r (namely, setting $A_t(r, i, x) := M_{i,x}(A_{t-1})$ for all i, x). But in the general case, the "row" labeled by $R = (r_1, \ldots, r_\ell)$ consists of all triples $\langle R', i, x\rangle$ such that $R' = \left(r_1, \ldots, r_\ell, r'_{\ell+1}, \ldots, r'_L\right)$ for some $L \geq \ell$ and $r'_{\ell+1}, \ldots, r'_L$. Furthermore, we do not yet know how later iterations will affect this "row." So we should call a pair $\langle i, x\rangle$ "sensitive" to R, if there is *any* oracle A' such that (1) A' disagrees with A only in row R, and (2) $M_{i,x}(A') \neq M_{i,x}(A)$.

Fortunately, this new notion of sensitivity requires no significant change to the proof. Suppose that for every row R of the form $\left(r_1^*, \ldots, r_{\ell-1}^*, r_\ell\right)$ there exists an $\langle i, x\rangle$ that is sensitive to R. Then as before, there exists an $\langle i', x'\rangle$ that is sensitive to at least $2^{5 \cdot 2^\ell} / \left(2^{2\ell} 2^{2^\ell+1}\right) > 2^{3n}$ rows of that form. For each of those rows R, fix a change to R to which $\langle i', x'\rangle$ is sensitive. We thereby obtain a polynomial $Q(A)$ with the same properties as before—in particular, there exists a string $Y \in \{0,1\}^{2^{3n}}$ such that $Q\left(A^{(Y)}\right) \geq 2Q(A)$. ■

Let us make three remarks about Theorem 2. First, if we care about constants, it is clear that the advice r can be reduced to $3n + o(n)$ bits for a specific n, or $12n + o(n)$ for all n simultaneously. Presumably these bounds are not tight. Second, one can easily extend Theorem 2 to give an oracle relative to which $\mathsf{PE} = \mathsf{PTIME}\left(2^{O(n)}\right)$ has linear-size circuits, and hence $\mathsf{PEXP} \subset \mathsf{P/poly}$ by a padding argument. Third, Han, Hemaspaandra, and Thierauf [18] showed that $\mathsf{MA} \subseteq \mathsf{BPP_{path}} \subseteq \mathsf{PP}$. So in addition to implying the result of Buhrman, Fortnow, and Thierauf that MA has linear-size circuits relative to an oracle, Theorem 2 also yields the new result that $\mathsf{BPP_{path}}$ has linear-size circuits relative to an oracle.

In Appendices 8 and 9, we will explain how the techniques of Theorem 2 can be used to prove several other results. In particular, in Appendix 8 we give relativized worlds where $\mathsf{P^{NP}} = \mathsf{PEXP}$ and $\oplus\mathsf{P} = \mathsf{PEXP}$, and in Appendix 9 we generalize the result of Beigel [8] that no small perceptron solves the ODDMAXBIT problem.

3. Quantum Circuit Lower Bounds

In this section we show, by a nonrelativizing argument, that PP does not have circuits of size n^k, not even quantum circuits with quantum advice. We first show that $\mathsf{P^{PP}}$ does not have quantum circuits of size n^k, by a direct diagonalization argument. Our argument will use the following lemma of Aaronson [1].

Lemma 3 ("Almost As Good As New Lemma") *Suppose a two-outcome measurement of a mixed quantum state ρ yields outcome 0 with probability $1 - \varepsilon$. Then after the measurement, we can recover a state $\widetilde{\rho}$ such that $\left\|\widetilde{\rho} - \rho\right\|_{\mathrm{tr}} \leq \sqrt{\varepsilon}$.*

(Recall that the trace distance $\left\|\rho - \sigma\right\|_{\mathrm{tr}}$ between two mixed states ρ and σ is the maximum bias with which those states can be distinguished via a single measurement. In particular, trace distance satisfies the triangle inequality.)

Theorem 4 $\mathsf{P^{PP}}$ *does not have quantum circuits of size n^k for any fixed k. Furthermore, this holds even if the circuits can use quantum advice.*

Proof. For simplicity, let us first explain why $\mathsf{P^{PP}}$ does not have *classical* circuits of size n^k. Fix an input length n, and let $x_1, \ldots, x_{2^n}$ be a lexicographic ordering of n-bit strings. Also, let $\mathcal{C}$ be the set of all circuits of size n^k, and let $\mathcal{C}_t \subseteq \mathcal{C}$ be the subset of circuits in $\mathcal{C}$ that correctly decide the first t inputs $x_1, \ldots, x_t$. Then we define the language $L \cap \{0,1\}^n$ by the following iterative procedure. First, if at least half of the circuits in $\mathcal{C}$ accept x_1, then set $x_1 \notin L$, and otherwise set $x_1 \in L$. Next, if at least half of the circuits in $\mathcal{C}_1$ accept x_2, then set $x_2 \notin L$, and otherwise set $x_2 \in L$. In general, let $N = \lceil \log_2 |\mathcal{C}'| \rceil + 1$. Then for all $t < N$, if at least half of the circuits in $\mathcal{C}_t$ accept x_{t+1}, then set $x_{t+1} \notin L$, and otherwise set $x_{t+1} \in L$. Finally, set $x_t \notin L$ for all $t > N$.

It is clear that the resulting language L is in P^{PP}. Given an input x_t, we just reject if $t > N$, and otherwise call the PP oracle t times, to decide if $x_i \in L$ for each $i \in \{1, \ldots, t\}$. Note that, once we know $x_1, \ldots, x_i$, we can decide in polynomial time whether a given circuit belongs to $\mathcal{C}_i$, and can therefore decide in PP whether the majority of circuits in $\mathcal{C}_i$ accept or reject x_{i+1}. On the other hand, our construction guarantees that $|\mathcal{C}_{t+1}| \leq |\mathcal{C}_t| / 2$ for all $t < N$. Therefore $|\mathcal{C}_N| \leq |\mathcal{C}| / 2^N = 1/2$, which means that $\mathcal{C}_N$ is empty, and hence no circuit in $\mathcal{C}$ correctly decides $x_1, \ldots, x_N$.

The above argument extends naturally to quantum circuits. Let $\mathcal{C}$ be the set of all quantum circuits of size n^k, over a basis of (say) Hadamard and Toffoli gates.[12] (Note that these circuits need not be bounded-error.) Then the first step is to amplify each circuit $C \in \mathcal{C}$ a polynomial number times, so that if C's initial error probability was at most $1/3$, then its new error probability is at most (say) 2^{-10n}. Let $\mathcal{C}'$ be the resulting set of amplified circuits. Now let $|\psi_0\rangle$ be a uniform superposition over all descriptions of circuits in $\mathcal{C}'$, together with an "answer register" that is initially set to $|0\rangle$:

$$|\psi_0\rangle := \frac{1}{\sqrt{|\mathcal{C}'|}} \sum_{C \in \mathcal{C}'} |C\rangle \, |0\rangle \, .$$

For each input $x_t \in \{0,1\}^n$, let U_t be a unitary transformation that maps $|C\rangle \, |0\rangle$ to $|C\rangle \, |C(x_t)\rangle$ for each $C \in \mathcal{C}'$, where $|C(x_t)\rangle$ is the output of C on input x_t. (In general, $|C(x_t)\rangle$ will be a superposition of $|0\rangle$ and $|1\rangle$.) To implement U_t, we simply simulate running C on x_t, and then run the simulation in reverse to uncompute garbage qubits.

Let $N = \lceil \log_2 |\mathcal{C}'| \rceil + 2$. Also, given an input x_t, let $L(x_t) = 1$ if $x_t \in L$ and $L(x_t) = 0$ otherwise. Fix $t < N$, and suppose by induction that we have already set $L(x_i)$ for all $i \leq t$. Then we will use the following quantum algorithm, called $\mathcal{A}_t$, to set $L(x_{t+1})$.

```
Set |ψ⟩ := |ψ_0⟩
For i := 1 to t
    Set |ψ⟩ := U_i |ψ⟩
    Measure the answer register
    If the measurement outcome
        is not L(x_i), then FAIL
Next i
Set |ψ⟩ := U_{t+1} |ψ⟩
Measure the answer register
```

Say that $\mathcal{A}_t$ succeeds if it outputs $L(x_i)$ for all $x_1, \ldots, x_t$. *Conditioned* on $\mathcal{A}_t$ succeeding, if the final measurement yields the outcome $|1\rangle$ with probability at least $1/2$, then set $L(x_{t+1}) := 0$, and otherwise set $L(x_{t+1}) := 1$. Finally, set $L(x_t) := 0$ for all $t > N$.

By a simple extension of the result $\mathsf{BQP} \subseteq \mathsf{PP}$ due to Adleman, DeMarrais, and Huang [3], Aaronson [2] showed that polynomial-time quantum computation with postselected measurement can be simulated in PP (indeed the two are equivalent; that is, PostBQP = PP). In particular, a PP machine can simulate the postselected quantum algorithm $\mathcal{A}_t$ above, and thereby decide whether the final measurement will yield $|0\rangle$ or $|1\rangle$ with greater probability, conditioned on all previous measurements having yielded the correct outcomes. It follows that $L \in \mathsf{P}^{\mathsf{PP}}$.

On the other hand, suppose by way of contradiction that there exists a quantum circuit $C \in \mathcal{C}'$ that outputs $L(x_t)$ with probability at least $1 - 2^{-10n}$ for all t. Then the probability that C succeeds on $x_1, \ldots, x_N$ *simultaneously* is at least (say) 0.9, by Lemma 3 together with the triangle inequality. Hence the probability that $\mathcal{A}_t$ succeeds on $x_1, \ldots, x_N$ is at least $0.9/|\mathcal{C}'|$. Yet by construction, $\mathcal{A}_t$ succeeds with probability at most $1/2^t$, which is less than $0.9/|\mathcal{C}'|$ when $t = N - 1$. This yields the desired contradiction.

Finally, to incorporate quantum advice of size $s = n^k$, all we need to do is add an s-qubit "quantum advice register" to $|\psi_0\rangle$, which U_t's can use when simulating the circuits. We initialize this advice register to the maximally mixed state on s qubits. The key fact (see [1] for example) is that, whatever the "true" advice state $|\phi\rangle$, we can decompose the maximally mixed state into

$$\frac{1}{2^s} \sum_{j=1}^{2^s} |\phi_j\rangle \langle \phi_j| \, ,$$

where $|\phi_1\rangle, \ldots, |\phi_{2^s}\rangle$ form an orthonormal basis and $|\phi_1\rangle = |\phi\rangle$. By linearity, we can then track the evolution of each of these 2^s components independently. So the previous argument goes through as before, if we set $N = \lceil \log_2 |\mathcal{C}'| \rceil + s + 2$. (Note that we are assuming the advice states are suitably amplified, which increases the running time of $\mathcal{A}_t$ by at most a polynomial factor.) ■

Similarly, for all time-constructible functions $f(n) \leq 2^n$, one can show that the class $\mathsf{DTIME}(f(n))^{\mathsf{PP}}$ does not have quantum circuits of size $f(n)/n^2$. So for example, E^{PP} requires quantum circuits of exponential size.

Having shown a quantum circuit lower bound for P^{PP}, we now bootstrap our way down to PP. To do so, we use the following "quantum Karp-Lipton theorem" (or more precisely, "quantum LFKN theorem"). Here BQP/poly is BQP with polynomial-size classical advice, BQP/qpoly is BQP with polynomial-size quantum advice, QMA is like MA but with quantum verifiers and quantum witnesses, and QCMA is like MA but with quantum verifiers and *classical* witnesses. Also, recall that the counting hierarchy CH is

[12] Shi [37] showed that this basis is universal. Any finite, universal set of gates with rational amplitudes would work equally well.

the union of PP, $\mathsf{PP}^{\mathsf{PP}}$, $\mathsf{PP}^{\mathsf{PP}^{\mathsf{PP}}}$, and so on.

Theorem 5 *If* $\mathsf{PP} \subset \mathsf{BQP/poly}$ *then* $\mathsf{QCMA} = \mathsf{PP}$, *and indeed* CH *collapses to* QCMA. *Likewise, if* $\mathsf{PP} \subset \mathsf{BQP/qpoly}$ *then* CH *collapses to* QMA.

Proof. Let L be a language in CH. It is clear that we could decide L in quantum polynomial time, if we were given polynomial-size quantum circuits for a PP-complete language such as MAJSAT. For Fortnow and Rogers [16] showed that BQP is "low" for PP; that is, $\mathsf{PP}^{\mathsf{BQP}} = \mathsf{PP}$. So we could use the quantum circuits for MAJSAT to collapse $\mathsf{PP}^{\mathsf{PP}}$ to $\mathsf{PP}^{\mathsf{BQP}} = \mathsf{PP}$ to BQP, and similarly for all higher levels of CH.

Assume $\mathsf{PP} \subset \mathsf{BQP/poly}$; then clearly $\mathsf{P}^{\#\mathsf{P}} = \mathsf{P}^{\mathsf{PP}}$ is contained in $\mathsf{BQP/poly}$ as well. So in QCMA we can do the following: first guess a bounded-error quantum circuit C for computing the permanent of a $\operatorname{poly}(n) \times \operatorname{poly}(n)$ matrix over a finite field $\mathbb{F}_p$, for some prime $p = \Theta(\operatorname{poly}(n))$. (For convenience, here $\operatorname{poly}(n)$ means "a sufficiently large polynomial depending on L.") Then verify that with $1 - o(1)$ probability, C works on at least a $1 - 1/\operatorname{poly}(n)$ fraction of matrices. To do so, simply simulate the interactive protocol for the permanent due to Lund, Fortnow, Karloff, and Nisan [26], but with C in place of the prover. Next, use the random self-reducibility of the permanent to generate a new circuit C' that, with $1 - o(1)$ probability, is correct on *every* $\operatorname{poly}(n) \times \operatorname{poly}(n)$ matrix over $\mathbb{F}_p$. Since PERMANENT is $\#\mathsf{P}$-complete over all fields of characteristic $p \neq 2$ [42], we can then use C' to decide MAJSAT instances of size $\operatorname{poly}(n)$, and therefore the language L as well.

The case $\mathsf{PP} \subset \mathsf{BQP/qpoly}$ is essentially identical, except that in QMA we guess a quantum circuit with quantum advice. That quantum advice states cannot be reused indefinitely does not present a problem here: we simply guess a boosted circuit, or else $\operatorname{poly}(n)$ copies of the original circuit. ■

By combining Theorems 4 and 5, we immediately obtain the following.

Corollary 6 PP *does not have quantum circuits of size* n^k *for any fixed* k, *not even quantum circuits with quantum advice.*

Proof. Suppose by contradiction that PP had such circuits. Then certainly $\mathsf{PP} \subset \mathsf{BQP/qpoly}$, so $\mathsf{QMA} = \mathsf{PP} = \mathsf{P}^{\mathsf{PP}} = \mathsf{CH}$ by Theorem 5. But P^{PP} does *not* have such circuits by Theorem 4, and therefore neither does PP. ■

More generally, for all $f(n) \leq 2^n$ we find that $\mathsf{PTIME}(f(f(n)))$ requires quantum circuits of size approximately $f(n)$. For example, PEXP requires quantum circuits of "half-exponential" size.

Finally, we point out a quantum analogue of Buhrman, Fortnow, and Thierauf's classical nonrelativizing separation [11].

Theorem 7 $\mathsf{QCMA}_{\mathsf{EXP}} \not\subset \mathsf{BQP/poly}$, *and* $\mathsf{QMA}_{\mathsf{EXP}} \not\subset \mathsf{BQP/qpoly}$.

Proof. Suppose by contradiction that $\mathsf{QCMA}_{\mathsf{EXP}} \subset \mathsf{BQP/poly}$. Then clearly $\mathsf{EXP} \subset \mathsf{BQP/poly}$ as well. Babai, Fortnow, and Lund [6] showed that any language in EXP has a two-prover interactive protocol where the provers are in EXP. We can simulate such a protocol in QCMA as follows: first guess (suitably amplified) $\mathsf{BQP/poly}$ circuits computing the provers' strategies. Then simulate the provers and verifier, and accept if and only if the verifier accepts. It follows that $\mathsf{EXP} = \mathsf{QCMA}$, and therefore $\mathsf{QCMA} = \mathsf{P}^{\mathsf{PP}}$ as well. So by padding, $\mathsf{QCMA}_{\mathsf{EXP}} = \mathsf{EXP}^{\mathsf{PP}}$. But we know from Theorem 4 that $\mathsf{EXP}^{\mathsf{PP}} \not\subset \mathsf{BQP/poly}$, which yields the desired contradiction. The proof that $\mathsf{QMA}_{\mathsf{EXP}} \not\subset \mathsf{BQP/qpoly}$ is essentially identical, except that we guess quantum circuits with quantum advice. ■

One can strengthen Theorem 7 to show that $\mathsf{QMA}_{\mathsf{EXP}}$ requires quantum circuits of half-exponential size. However, in contrast to the case for PEXP, here the bound does not scale down to QMA. Indeed, it turns out that the smallest f for which we get *any* superlinear circuit size lower bound for $\mathsf{QMATIME}(f(n))$ is itself half-exponential.

4. The Oracle for $\mathsf{BPP}_{||}^{\mathsf{NP}}$

In this section we construct an oracle relative to which $\mathsf{BPP}_{||}^{\mathsf{NP}}$ has linear-size circuits.

Theorem 8 *There exists an oracle relative to which* $\mathsf{BPP}_{||}^{\mathsf{NP}}$ *has linear-size circuits.*

Proof. As in Theorem 2, we first give an oracle A that works for a specific value of n. Let $M_1, M_2, \ldots$ be an enumeration of "syntactic" $\mathsf{BPTIME}\left(n^{\log n}\right)_{||}^{\mathsf{NP}}$ machines, where syntactic means not necessarily satisfying the promise. Then it suffices to simulate $M_1, \ldots, M_n$. We assume without loss of generality that only the NP oracle (not the M_i's themselves) query A, and that each NP call is actually an $\mathsf{NTIME}(n)$ call (so in particular, it involves at most $n^{\log n}$ queries to A). Let $M_{i,x,z}(A)$ be a Boolean function that equals 1 if M_i accepts on input $x \in \{0,1\}^n$, random string $z \in \{0,1\}^{n^{\log n}}$, and oracle A, and 0 otherwise. Then let $p_{i,x}(A) := \mathrm{EX}_z[M_{i,x,z}(A)]$ be the probability that M_i accepts x.

The oracle A will consist of 2^{3n} rows and $n2^n$ columns, with each row labeled by $r \in \{0,1\}^{3n}$, and each column labeled by an $\langle i, x\rangle$ pair where $i \in \{1, \ldots, n\}$ and $x \in \{0,1\}^n$. We will construct A via an iterative procedure $\mathcal{P}$. Initially A is empty (that is, $A(r,i,x) = 0$ for all r, i, x). Let A_t be the state of A after the t^{th} iteration. Then to *encode* a row r means to set $A_t(r,i,x) :=$

$\mathrm{round}\left(p_{i,x}\left(A_{t-1}\right)\right)$ for all i, x, where $\mathrm{round}\left(p\right) = 1$ if $p \geq 1/2$ and $\mathrm{round}\left(p\right) = 0$ if $p < 1/2$.

Call an $\langle i, x\rangle$ pair *sensitive* to row r, if encoding r would change $p_{i,x}\left(A\right)$ by at least $1/6$. Then $\mathcal{P}$ consists entirely of repeating the following two steps, for $t = 1, 2, 3 \ldots$:

(1) If there exists an r to which no $\langle i, x\rangle$ is sensitive, then encode r and halt.

(2) Otherwise, by a counting argument, there exists a pair $\langle j, y\rangle$ that is sensitive to at least $N = 2^{3n}/\left(n2^n\right)$ rows, call them $r_1, \ldots, r_N$. Let $A^{(k)}$ be the oracle obtained by starting from A and then encoding r_k. Choose an integer $k \in \{1, \ldots, N\}$ (we will specify how later), and set $A_t := A_{t-1}^{(k)}$.

Suppose $\mathcal{P}$ halts after t iterations, and let r be the row encoded by step (1). Then by assumption, $\left|p_{i,x}\left(A_t\right) - p_{i,x}\left(A_{t-1}\right)\right| < 1/6$ for all i, x. So in particular, if $p_{i,x}\left(A_t\right) \geq 2/3$ then $p_{i,x}\left(A_{t-1}\right) > 1/2$ and therefore $A_t\left(r, i, x\right) = 1$. Likewise, if $p_{i,x}\left(A_t\right) \leq 1/3$ then $p_{i,x}\left(A_{t-1}\right) < 1/2$ and therefore $A_t\left(r, i, x\right) = 0$. It follows that any valid $\mathsf{BPTIME}\left(n^{\log n}\right)_{||}^{\mathsf{NP}}$ machine in $\{M_1, \ldots, M_n\}$ has linear-size circuits relative to A_t—since we can just hardwire $r \in \{0,1\}^{2n}$ into the circuits.

It remains only to show that $\mathcal{P}$ halts after a finite number of steps, for some choice of k's. Given an input x, random string z, and oracle A, let $S_{i,x,z}\left(A\right)$ be the set of NP queries made by M_i that accept. Then we will use

$$W\left(A\right) := \sum_{i,x} \operatorname*{EX}_{z}\left[\left|S_{i,x,z}\left(A\right)\right|\right]$$

as our progress measure. Since each M_i can query the NP oracle at most $n^{\log n}$ times, clearly $0 \leq \left|S_{i,x,z}\left(A\right)\right| \leq n^{\log n}$ for all i, x, z, and therefore

$$0 \leq W\left(A\right) \leq n2^n \cdot n^{\log n}$$

for all A. On the other hand, we claim that whenever step (2) is executed, if $k \in \{1, \ldots, N\}$ is chosen uniformly at random then

$$\operatorname*{EX}_{k}\left[W\left(A^{(k)}\right)\right] \geq W\left(A\right) + \frac{1}{6} - 2^{-n+o(n)}.$$

So in step (2), we should simply choose k to maximize $W\left(A^{(k)}\right)$. For we will then have $W\left(A_t\right) \geq \left(1/6 - 2^{-n+o(n)}\right) t$ for all t, from which it follows that $\mathcal{P}$ halts after at most

$$\frac{n2^n \cdot n^{\log n}}{1/6 - 2^{-n+o(n)}} = 2^{n+o(n)}$$

iterations.

We now prove the claim. Observe that for each accepting NP query $q \in S_{i,x,z}\left(A\right)$, there are at most $n^{\log n}$ rows r_k such that encoding r_k would cause $q \notin S_{i,x,z}\left(A^{(k)}\right)$. For to change q's output from 'accept' to 'reject,' we would have to eliminate (say) the lexicographically first accepting path of the NP oracle, and that path can depend on at most $n^{\log n}$ rows of A. Hence by the union bound, for all i, x, z, A we have

$$\begin{aligned}
&\Pr_{k}\left[S_{i,x,z}\left(A\right) \not\subset S_{i,x,z}\left(A^{(k)}\right)\right] \\
&\quad\leq \sum_{q \in S_{i,x,z}(A)} \Pr_{k}\left[q \notin S_{i,x,z}\left(A^{(k)}\right)\right] \\
&\quad\leq \left|S_{i,x,z}\left(A\right)\right| \frac{n^{\log n}}{N} \\
&\quad\leq \frac{n^{2\log n}}{2^{3n}/\left(n2^n\right)} \\
&\quad= 2^{-2n+o(n)}.
\end{aligned}$$

So in particular, for all i, x, A,

$$\begin{aligned}
&\operatorname*{EX}_{k,z}\left[\left|S_{i,x,z}\left(A^{(k)}\right)\right|\right] \\
&\quad\geq \left|S_{i,x,z}\left(A\right)\right| \cdot \Pr_{k,z}\left[\left|S_{i,x,z}\left(A^{(k)}\right)\right| \geq \left|S_{i,x,z}\left(A\right)\right|\right] \\
&\quad\geq \left|S_{i,x,z}\left(A\right)\right| \left(1 - 2^{-2n+o(n)}\right)
\end{aligned}$$

On the other hand, by assumption there exists a pair $\langle j, y\rangle$ that is sensitive to row r_k for every $k \in \{1, \ldots, N\}$. Furthermore, given y and z, the output $M_{j,y,z}\left(A\right)$ of M_j is a function of the NP oracle responses $S_{j,y,z}\left(A\right)$, and can change only if $S_{j,y,z}\left(A\right)$ changes. Therefore

$$\begin{aligned}
&\Pr_{k,z}\left[S_{j,y,z}\left(A^{(k)}\right) \neq S_{j,y,z}\left(A\right)\right] \\
&\quad\geq \Pr_{k,z}\left[M_{j,y,z}\left(A^{(k)}\right) \neq M_{j,y,z}\left(A\right)\right] \\
&\quad\geq \frac{1}{6}.
\end{aligned}$$

So by the union bound,

$$\begin{aligned}
&\Pr_{k,z}\left[\left|S_{j,y,z}\left(A^{(k)}\right)\right| > \left|S_{j,y,z}\left(A\right)\right|\right] \\
&\quad\geq \Pr_{k,z}\left[S_{j,y,z}\left(A^{(k)}\right) \neq S_{j,y,z}\left(A\right)\right] \\
&\qquad - \Pr_{k,z}\left[S_{j,y,z}\left(A\right) \not\subset S_{j,y,z}\left(A^{(k)}\right)\right] \\
&\quad\geq \frac{1}{6} - 2^{-2n+o(n)}.
\end{aligned}$$

Putting it all together,

$$\begin{aligned}
&\mathop{\mathrm{EX}}_{k}\left[W\left(A^{(k)}\right)\right]\\
&\quad=\sum_{i,x}\mathop{\mathrm{EX}}_{k,z}\left[\left|S_{i,x,z}\left(A^{(k)}\right)\right|\right]\\
&\quad\geq\frac{1}{6}-2^{-2n+o(n)}+\sum_{i,x}\left|S_{i,x,z}\left(A\right)\right|\left(1-2^{-2n+o(n)}\right)\\
&\quad=\frac{1}{6}-2^{-2n+o(n)}+\left(1-2^{-2n+o(n)}\right)W\left(A\right)\\
&\quad=W\left(A\right)+\frac{1}{6}-2^{-n+o(n)},
\end{aligned}$$

which completes the claim.

To handle all values of n simultaneously, we use exactly the same trick as in Theorem 2. That is, we replace r by an ℓ-tuple $R=(r_1,\ldots,r_\ell)$ where $r_\ell\in\{0,1\}^{3\cdot 2^\ell}$; define the "row" $\mathcal{R}_\ell$ to consist of all triples $\langle R'_L,i,x\rangle$ such that $L\geq\ell$ and $r'_h=r_h$ for all $h\leq\ell$; and call the pair $\langle i,x\rangle$ "sensitive" to row $\mathcal{R}_\ell$ if there is any oracle A' that disagrees with A only in $\mathcal{R}_\ell$, such that $\left|p_{i,x}\left(A'\right)-p_{i,x}\left(A\right)\right|\geq 1/6$. We then run the procedure $\mathcal{P}$ repeatedly to encode $r_1,r_2,\ldots$, where "encoding" r_ℓ means setting $A_t\left(R_\ell,i,x\right):=\operatorname{round}\left(p_{i,x}\left(A_{t-1}\right)\right)$ for all $n\leq 2^\ell$, $i\in\{1,\ldots,n\}$, and $x\in\{0,1\}^n$. The rest of the proof goes through as before. ■

Let us make seven remarks about Theorem 8.

(1) Since we never needed the BPP promise, it is clear that Theorem 8 generalizes to $\mathsf{PromiseBPP}_{||}^{\mathsf{NP}}$.

(2) A corollary of Theorem 8 is that any Karp-Lipton collapse to $\mathsf{BPP}_{||}^{\mathsf{NP}}$ would require nonrelativizing techniques. For relative to the oracle A from the theorem, we have $\mathsf{NP}\subseteq\mathsf{BPP}_{||}^{\mathsf{NP}}\subset\mathsf{P/poly}$. On the other hand, if $\mathsf{PH}^A=\mathsf{BPP}_{||}^{\mathsf{NP}^A}$, then $\mathsf{BPP}_{||}^{\mathsf{NP}^A}$ would not have linear-size circuits by Kannan's Theorem [20] (which relativizes), thereby yielding a contradiction.

(3) If we care about constants, we can reduce the advice r to $2n+o(n)$ bits for a specific n, or $8n+o(n)$ for all n simultaneously.

(4) As with Theorem 2, one can easily modify Theorem 8 to give a relativized world where $\mathsf{BPEXP}_{||}^{\mathsf{NP}}\subset\mathsf{P/poly}$. Thus, Theorem 8 provides an alternate generalization of the result of Buhrman, Fortnow, and Thierauf [11] that $\mathsf{MA}_{\mathsf{EXP}}\subset\mathsf{P/poly}$ relative to an oracle.

(5) Since $\mathsf{BPP}_{\mathsf{path}}\subseteq\mathsf{BPP}_{||}^{\mathsf{NP}}$ (as is not hard to show using approximate counting), Theorem 8 also provides an alternate proof that $\mathsf{BPP}_{\mathsf{path}}$ has linear-size circuits relative to an oracle.

(6) Completely analogously to Theorem 12, one can modify Theorem 8 to give oracles relative to which $\mathsf{P}^{\mathsf{NP}}=\mathsf{BPEXP}_{||}^{\mathsf{NP}}$ and $\oplus\mathsf{P}=\mathsf{BPEXP}_{||}^{\mathsf{NP}}$.

(7) For any function f, the construction of Theorem 8 actually yields an oracle relative to which $\mathsf{BPP}^{\mathsf{NP}[f(n)]}$ (that is, BPP with $f(n)$ adaptive NP queries) has circuits of size $O(n+f(n))$. For clearly we can simulate $f(n)$ adaptive queries using $2^{f(n)}$ nonadaptive queries. We then repeat Theorem 8 with the bound $0\leq W(A)\leq n2^n\cdot 2^{f(n)}$.

5. Black-Box Learning in Algorithmica

"Algorithmica" is one of Impagliazzo's five possible worlds [19], the world in which $\mathsf{P}=\mathsf{NP}$. In this section we show that in Algorithmica, black-box learning of Boolean circuits is possible in $\mathsf{P}_{||}^{\mathsf{NP}}$. Let us first define what we mean by black-box learning.

Definition 9 *Say that black-box learning is possible in a complexity class $\mathcal{C}$ if the following holds. There exists a $\mathcal{C}$ machine M such that, for all Boolean functions $f:\{0,1\}^n\to\{0,1\}$ with circuit complexity at most $s(n)$, the machine M^f outputs a circuit for f given $\langle 0^n,0^{s(n)}\rangle$ as input. Also, M has approximation ratio $\alpha(n)$ if for all f, any circuit output by M has size at most $s(n)\,\alpha(n)$.*

The above definition is admittedly somewhat vague, but for most natural complexity classes $\mathcal{C}$ it is clear how to make it precise. Firstly, by "$\mathcal{C}$ machine" we really mean "$\mathcal{FC}$ machine," where $\mathcal{FC}$ is the function version of $\mathcal{C}$. Secondly, for semantic classes, we do not care if the machine violates the promise on inputs not of the form $\langle 0^n,0^{s(n)}\rangle$, or oracles f that do not have circuit complexity at most $s(n)$.

Let us give a few examples. First, almost by definition, black-box learning is possible in Σ_2^p with approximation ratio 1. Second, as pointed out by Umans [41], the result of Bshouty et al. [9] implies that black-box learning is possible in $\mathsf{ZPP}^{\mathsf{NP}}$, with approximation ratio $O(n/\log n)$. Third, under plausible assumptions, black-box learning is possible in P^{NP} with approximation ratio $O(n/\log n)$.[13] Fourth, if E requires MAJSAT-oracle circuits of size $2^{\Omega(n)}$, then black-box learning is possible in PP with approximation ratio 1. For this assumption implies that $\mathsf{PP}=\mathsf{BP}\cdot\mathsf{PP}$, and hence that $\Sigma_2^p\subseteq\mathsf{PP}$ by Toda's theorem.

On the other hand:

Proposition 10 *Black-box learning is impossible in* NP*, or for that matter in* AM*,* IP*, or* MIP*.*

Proof. Suppose there are two possibilities: either f is the identically zero function, or else f is a point function (that is, there exists a y such that $f(x)=1$ if and only if $x=y$). In both cases $s(n)=O(n)$. But since the verifier has

[13] For it follows from a general result of Klivans and van Melkebeek [22] that if E requires SAT-oracle circuits of size $2^{\Omega(n)}$, then $\mathsf{P}^{\mathsf{NP}}=\mathsf{ZPP}^{\mathsf{NP}}$. (Here a SAT-oracle circuit is a circuit with oracle access to SAT.) Furthermore, the derandomization result of [22] is "black-box respecting," in the sense that if the $\mathsf{ZPP}^{\mathsf{NP}}$ machine doesn't "cheat" by exploiting the internal structure of the circuit to be learned, then neither does its P^{NP} simulation.

only oracle access to f, it is obvious that no polynomially-bounded sequence of messages from the prover(s) could convince the verifier that f is identically zero. We omit the details, which were worked out by Fortnow and Sipser [17]. ■

We now prove the main result.

Theorem 11 *If* $\mathsf{P} = \mathsf{NP}$*, then black-box learning is possible in* $\mathsf{P}_{||}^{\mathsf{NP}}$ *(indeed, with approximation ratio* 1*).*

Proof. Fix n, and suppose $f : \{0,1\}^n \rightarrow \{0,1\}$ has circuits of size $s = s(n)$. Let $\mathcal{B}$ be the set of all circuits of size s, so that $|\mathcal{B}| = s^{O(s)}$. Also, say that a circuit $C \in \mathcal{B}$ *succeeds* on input $x \in \{0,1\}^n$ if $C(x) = f(x)$, and *fails* otherwise. Then given a list of inputs $X = (x_1, x_2, \ldots)$, let $\mathcal{B}(X)$ be the set of circuits in $\mathcal{B}$ that succeed on every $x \in X$.

For the remainder of the proof, let $X_t = (x_1, \ldots, x_t)$ be a list of t inputs, and for all $0 \leq i < t$, let $X_i = (x_1, \ldots, x_i)$ be the prefix of X_t consisting of the first i inputs (so in particular, X_0 is the empty list). Then our first claim is that there exists an NP^f machine Q_t with the following behavior:

- If there exists an X_t such that $|\mathcal{B}(X_i)| \leq \frac{2}{3} |\mathcal{B}(X_{i-1})|$ for all $i \in \{1, \ldots, t\}$, then Q_t accepts.
- If for all X_t there exists an $i \in \{1, \ldots, t\}$ such that $|\mathcal{B}(X_i)| \geq \frac{3}{4} |\mathcal{B}(X_{i-1})|$, then Q_t rejects.

(As usual, if neither of the two stated conditions hold, then the machine can behave arbitrarily.)

In what follows, we can assume without loss of generality that t is polynomially bounded. For, since *some* circuit $C \in \mathcal{B}$ succeeds on every input, we must have $|\mathcal{B}(X_i)| \geq 1$ for all i. Therefore Q_t can accept only if $|\mathcal{B}| (3/4)^t \geq 1$, or equivalently if $t = O(s \log s)$.

Let $f(X_t) := (f(x_1), \ldots, f(x_t))$, and let z be a "witness string" consisting of X_t and $f(X_t)$. Then given z and $i \leq t$, we can easily decide whether a circuit C belongs to the set $\mathcal{B}(X_i)$: we simply check whether $C(x_j) = f(x_j)$ for all $j \leq i$. So by standard results on approximate counting due to Stockmeyer [39] and Sipser [38], we can approximate the cardinality $|\mathcal{B}(X_i)|$ in $\mathsf{BPP}^{\mathsf{NP}}$. More precisely, for all t, i there exists a $\mathsf{PromiseBPP}^{\mathsf{NP}}$ machine $M_{t,i}$ such that for all $z = \langle X_t, f(X_t) \rangle$:

- If $|\mathcal{B}(X_i)| \leq \frac{2}{3} |\mathcal{B}(X_{i-1})|$ then $M_{t,i}(z)$ accepts with probability at least $2/3$ (where the probability is over $M_{t,i}$'s internal randomness).
- If $|\mathcal{B}(X_i)| \geq \frac{3}{4} |\mathcal{B}(X_{i-1})|$ then $M_{t,i}(z)$ rejects with probability least $2/3$.

Now by the Sipser-Lautemann Theorem [38, 25], the assumption $\mathsf{P} = \mathsf{NP}$ implies that $\mathsf{PromiseP} = \mathsf{PromiseBPP}^{\mathsf{NP}}$ as well. So we can convert $M_{t,i}$ into a deterministic polynomial-time machine $M'_{t,i}$ such that for all z, if $|\mathcal{B}(X_i)| \leq \frac{2}{3} |\mathcal{B}(X_{i-1})|$ then $M'_{t,i}(z)$ accepts, while if $|\mathcal{B}(X_i)| \geq \frac{3}{4} |\mathcal{B}(X_{i-1})|$ then $M'_{t,i}(z)$ rejects.

Using $M'_{t,i}$, we can then rewrite Q_t as follows: "Does there exist a witness z, of the form $\langle X_t, f(X_t) \rangle$, such that $M'_{t,1}(z) \wedge \cdots \wedge M'_{t,t}(z)$?" This proves the claim, since the above query is clearly in NP^f.

To complete the theorem, we will need one other predicate $A_t(z, x)$, with the following behavior. For all $z = \langle X_t, f(X_t) \rangle$ and $x \in \{0,1\}^n$, if $\Pr_{C \in \mathcal{B}(X_t)} [C(x) = 1] \geq 2/3$ then $A_t(z, x)$ accepts, while if $\Pr_{C \in \mathcal{B}(X_t)} [C(x) = 0] \geq 2/3$ then $A_t(z, x)$ rejects.

It is clear that we can implement A_t in $\mathsf{PromiseBPP}^{\mathsf{NP}}$, again because of approximate counting and the ease of deciding membership in $\mathcal{B}(X_t)$. So by the assumption $\mathsf{P} = \mathsf{NP}$, we can also implement A_t in P.

Now let $C_{t,z}$ be the lexicographically first circuit $C \in \mathcal{B}$ such that $C(x) = A_t(z, x)$ for all $x \in \{0,1\}^n$. Notice that $A_t(z, x)$ is an *explicit procedure*: that is, we can evaluate it without recourse to the oracle for f. So given z, we can find $C_{t,z}$ in $\Delta_3^p = \mathsf{P}^{\mathsf{NP}^{\mathsf{NP}}}$, and hence also in P.

Let t^* be the maximum t for which Q_t accepts, and let $z = \langle X_{t^*}, f(X_{t^*}) \rangle$ be any accepting witness for Q_{t^*}. Then for all $x \in \{0,1\}^n$, we have

$$\Pr_{C \in \mathcal{B}(X_{t^*})} [C(x) = f(x)] \geq \frac{2}{3}.$$

For otherwise the sequence $(x_1, \ldots, x_{t^*}, x)$ would satisfy Q_{t^*+1}, thereby contradicting the maximality of t^*. An immediate corollary is that $A_{t^*}(z, x) = f(x)$ for all $x \in \{0,1\}^n$. Hence $C_{t^*,z}$ is the lexicographically first circuit for f, independently of the particular accepting witness z.

The $\mathsf{P}_{||}^{\mathsf{NP}^f}$ learning algorithm now follows easily. For all $t = O(s \log s)$, the algorithm submits the query Q_t to the NP oracle. It also submits the following query, called $R_{t,j}$, for all $t = O(s \log s)$ and $j = O(s \log s)$: "Does there exist a witness $z = \langle X_t, f(X_t) \rangle$ satisfying Q_t, such that the j^{th} bit in the description of $C_{t,z}$ is a 1?"

Using the responses to the Q_t's, the algorithm then determines t^*. Finally it reads a description of $C_{t^*,z}$ off the responses to the $R_{t^*,j}$'s. ■

Consider the following question: "why hasn't anyone managed to show a Karp-Lipton collapse to $\mathsf{P}_{||}^{\mathsf{NP}}$ or $\mathsf{BPP}_{||}^{\mathsf{NP}}$?" We might hope to answer this question by proving a "metaresult," stating that any such collapse would require non-black-box techniques. But Theorem 11 yields the "metametaresult" that we *can't* show such a metaresult, without also showing that $\mathsf{P} \neq \mathsf{NP}$!

As a final note, one corollary of Theorem 11 is that if $\mathsf{P} = \mathsf{NP}$, then black-box learning is possible in NP/log.

For since the $\mathsf{P}_{||}^{\mathsf{NP}}$ algorithm of Theorem 11 does not take any input, we simply count how many of its NP queries return a positive answer, and then feed that number as advice to the NP/log machine.

6. Open Problems

The main open problem is, of course, to prove better nonrelativizing lower bounds. For example, can we show that $\mathsf{BPP}_{||}^{\mathsf{NP}}$ does not have linear-size circuits? To do so, we would presumably need a nonrelativizing technique that applies directly to the polynomial hierarchy, without requiring the full strength of #P. Arora, Impagliazzo, and Vazirani [4] argue that "local checkability," as used for example in the PCP Theorem, constitutes such a technique (though see Fortnow [14] for a contrary view). For us, the relevant question is not which techniques are "truly" nonrelativizing, but simply which ones lead to lower bounds!

Here are a few other problems.

(1) Can we show that $\mathsf{P}^{\mathsf{NP}} \neq \mathsf{PEXP}$? If so, then we would obtain perhaps the first nonrelativizing separation of uniform complexity classes that does not follow immediately from a collapse such as $\mathsf{IP} = \mathsf{PSPACE}$ or $\mathsf{MIP} = \mathsf{NEXP}$.

(2) Can we show that PEXP requires circuits of exponential size, rather than just half-exponential?

(3) As mentioned in Section 1.2, Bshouty et al.'s algorithm does not find a *minimal* circuit for a Boolean function f, but only a circuit within an $O(n/\log n)$ factor of minimal.[14] Can we improve this approximation ratio, or alternatively, show that doing so would require nonrelativizing techniques?

(4) Is black-box learning possible in $\mathsf{P}_{||}^{\mathsf{NP}}$ or $\mathsf{ZPP}_{||}^{\mathsf{NP}}$, under some computational assumption that we actually believe (for example, a derandomization assumption)? Alternatively, can we show that black-box learning is *impossible* in $\mathsf{P}_{||}^{\mathsf{NP}}$ under some plausible computational assumption?

(5) Can we show that if $\mathsf{NP} \subseteq \mathsf{P/poly}$ then $\mathsf{PH} \subseteq \mathsf{PP}$? Of course, Theorem 2 implies that nonrelativizing techniques would be needed.

7. Acknowledgments

I am grateful to Lance Fortnow for telling me the problem of whether PP has linear-size circuits relative to an oracle, and for pointing out the implications of my oracle construction for perceptrons and for the relativized collapse of PEXP. I also thank Avi Wigderson for sponsoring the postdoc during which this work was done and for many enlightening conversations; the anonymous reviewers for their comments; and Boaz Barak, Dieter van Melkebeek, Sasha Razborov, Rahul Santhanam, Ronen Shaltiel, Luca Trevisan, Chris Umans, Umesh Vazirani, Hoeteck Wee, and Chris Wilson for helpful discussions and correspondence.

[14] Actually, the algorithm as we stated it gives an $O(n)$ approximation ratio, but we can improve it to $O(n/\log n)$ by replacing "at least a $1/3$ fraction" by "at least a $1/\operatorname{poly}(n)$ fraction."

References

[1] S. Aaronson. Limitations of quantum advice and one-way communication. *Theory of Computing*, 1:1–28, 2005. quant-ph/0402095.

[2] S. Aaronson. Quantum computing, postselection, and probabilistic polynomial-time. *Proc. Roy. Soc. London*, A461(2063):3473–3482, 2005. quant-ph/0412187.

[3] L. Adleman, J. DeMarrais, and M.-D. Huang. Quantum computability. *SIAM J. Comput.*, 26(5):1524–1540, 1997.

[4] S. Arora, R. Impagliazzo, and U. Vazirani. Relativizing versus nonrelativizing techniques: the role of local checkability. Manuscript, 1992.

[5] J. Aspnes, R. Beigel, M. Furst, and S. Rudich. The expressive power of voting polynomials. *Combinatorica*, 14(2):1–14, 1994.

[6] L. Babai, L. Fortnow, and C. Lund. Nondeterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1(1):3–40, 1991.

[7] T. Baker, J. Gill, and R. Solovay. Relativizations of the P=?NP question. *SIAM J. Comput.*, 4:431–442, 1975.

[8] R. Beigel. Perceptrons, PP, and the polynomial hierarchy. *Computational Complexity*, 4:339–349, 1994.

[9] N. H. Bshouty, R. Cleve, R. Gavaldà, S. Kannan, and C. Tamon. Oracles and queries that are sufficient for exact learning. *J. Comput. Sys. Sci.*, 52(3):421–433, 1996.

[10] H. Buhrman, S. Fenner, L. Fortnow, and L. Torenvliet. Two oracles that force a big crunch. *Computational Complexity*, 10(2):93–116, 2001.

[11] H. Buhrman, L. Fortnow, and T. Thierauf. Nonrelativizing separations. In *Proc. IEEE Complexity*, pages 8–12, 1998.

[12] J.-Y. Cai. $S_2^p \subseteq ZPP^{NP}$. In *Proc. IEEE FOCS*, pages 620–629, 2001.

[13] H. Ehlich and K. Zeller. Schwankung von Polynomen zwischen Gitterpunkten. *Mathematische Zeitschrift*, 86:41–44, 1964.

[14] L. Fortnow. The role of relativization in complexity theory. *Bulletin of the EATCS*, 52:229–244, February 1994.

[15] L. Fortnow and A. Klivans. NP with small advice. In *Proc. IEEE Complexity*, pages 228–234, 2005.

[16] L. Fortnow and J. Rogers. Complexity limitations on quantum computation. *J. Comput. Sys. Sci.*, 59(2):240–252, 1999. cs.CC/9811023.

[17] L. Fortnow and M. Sipser. Are there interactive protocols for co-NP languages? *Inform. Proc. Lett.*, 28:249–251, 1988.

[18] Y. Han, L. Hemaspaandra, and T. Thierauf. Threshold computation and cryptographic security. *SIAM J. Comput.*, 26(1):59–78, 1997.

[19] R. Impagliazzo. A personal view of average-case complexity. In *Proc. IEEE Complexity*, pages 134–147, 1995.

[20] R. Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55:40–56, 1982.

[21] R. M. Karp and R. J. Lipton. Turing machines that take advice. *Enseign. Math.*, 28:191–201, 1982.
[22] A. Klivans and D. van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM J. Comput.*, 31:1501–1526, 2002. Earlier version in ACM STOC 1999.
[23] J. Köbler and O. Watanabe. New collapse consequences of NP having small circuits. *SIAM J. Comput.*, 28(1):311–324, 1998.
[24] M. W. Krentel. The complexity of optimization problems. *J. Comput. Sys. Sci.*, 36(3):490–509, 1988.
[25] C. Lautemann. BPP and the polynomial hierarchy. *Inform. Proc. Lett.*, 17:215–217, 1983.
[26] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39:859–868, 1992.
[27] P. B. Miltersen, N. V. Vinodchandran, and O. Watanabe. Super-polynomial versus half-exponential circuit size in the exponential hierarchy. In *COCOON*, pages 210–220, 1999.
[28] M. Minsky and S. Papert. *Perceptrons (2nd edition)*. MIT Press, 1988. First appeared in 1968.
[29] N. Nisan and M. Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4(4):301–313, 1994.
[30] H. Nishimura and T. Yamakami. Polynomial time quantum computation with advice. *Inform. Proc. Lett.*, 90:195–204, 2003. ECCC TR03-059, quant-ph/0305100.
[31] W. J. Paul, N. Pippenger, E. Szemerédi, and W. T. Trotter. On determinism versus non-determinism and related problems. In *Proc. IEEE FOCS*, pages 429–438, 1983.
[32] A. A. Razborov and S. Rudich. Natural proofs. *J. Comput. Sys. Sci.*, 55(1):24–35, 1997.
[33] T. J. Rivlin and E. W. Cheney. A comparison of uniform approximations on an interval and a finite subset thereof. *SIAM J. Numerical Analysis*, 3(2):311–320, 1966.
[34] R. Shaltiel and C. Umans. Pseudorandomness for approximate counting and sampling. In *Proc. IEEE Complexity*, 2005. To appear.
[35] A. Shamir. IP=PSPACE. *J. ACM*, 39(4):869–877, 1992.
[36] C. Shannon. The synthesis of two-terminal switching circuits. *Bell System Technical Journal*, 28(1):59–98, 1949.
[37] Y. Shi. Both Toffoli and controlled-NOT need little help to do universal quantum computation. *Quantum Information and Computation*, 3(1):84–92, 2002. quant-ph/0205115.
[38] M. Sipser. A complexity theoretic approach to randomness. In *Proc. ACM STOC*, pages 330–335, 1983.
[39] L. J. Stockmeyer. The complexity of approximate counting. In *Proc. ACM STOC*, pages 118–126, 1983.
[40] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.
[41] C. Umans. *Approximability and Completeness in the Polynomial Hierarchy*. PhD thesis, UC Berkeley, 2000.
[42] L. G. Valiant. The complexity of computing the permanent. *Theoret. Comput. Sci.*, 8(2):189–201, 1979.
[43] L. G. Valiant and V. V. Vazirani. NP is as easy as detecting unique solutions. *Theoret. Comput. Sci.*, 47(3):85–93, 1986.
[44] N. Vereshchagin. On the power of PP. In *Proc. IEEE Complexity*, pages 138–143, 1992.
[45] N. V. Vinodchandran. A note on the circuit complexity of PP. ECCC TR04-056, 2004.
[46] C. B. Wilson. Relativized circuit complexity. *J. Comput. Sys. Sci.*, 31(2):169–181, 1985.

8. Appendix: A *Really* Big Crunch

By slightly modifying the construction of Theorem 2, we can resolve two other open questions of Fortnow.

Theorem 12

(i) There exists an oracle relative to which $\mathsf{P}^{\mathsf{NP}} = \mathsf{PEXP}$*, and indeed* $\mathsf{P}^{\mathsf{NP}} = \mathsf{P}^{\mathsf{NP}^{\mathsf{PEXP}}}$.

(ii) There exists an oracle relative to which $\oplus\mathsf{P} = \mathsf{PEXP}$.

Proof. For (i): in the oracle construction of Theorem 2 dealing with all n simultaneously, make the following simple change. Whenever a row R gets encoded, record the "current time" t as a prefix to that row. In other words, the oracle A will now take two kinds of queries: those of the form $\langle R, i, x \rangle$ as before, and those of the form $\langle R, j \rangle$ for an integer $j \geq 0$. Initially $A(R, j) = 0$ for all R, j. At any step of the iterative procedure, let t be the number of encoding steps that have already occurred. Then call the pair $\langle i, x \rangle$ "sensitive" to row R, if there exists an oracle A' such that (1) A' disagrees with A only in row R, (2) $M_{i,x}(A') \neq M_{i,x}(A)$, and (3) as we range over j, the $A'(R, j)$'s encode the binary expansion of $t+1$.

Clearly the proof of Theorem 2 still goes through with this change. For let $\ell = \lceil \log_2 n \rceil$. Then as before, whenever there does not exist a row R of the form $(r_1^*, \ldots, r_{\ell-1}^*, r_\ell)$ to which no $\langle i, x \rangle$ is sensitive, we can encode a subset of those rows so as to double $Q(A)$. Since $2^{-2^{O(n)}} \leq Q(A) \leq 2^{2^{O(n)}}$ for all A, this process will halt after at most $2^{O(n)}$ steps, meaning that t will never require more than $O(n)$ bits to represent. Indeed, this is true even if we are dealing with $\mathsf{PTIME}(2^n)$ machines, rather than $\mathsf{PTIME}(n^{\log n})$ machines.

Now consider a $\mathsf{PTIME}^A(2^n)$ machine M_i. We can simulate M_i in $\mathsf{DTIME}(n^2)^{\mathsf{NP}^A}$, as follows. Given an input $x \in \{0,1\}^n$, first find the unique row $R = (r_1, \ldots, r_{\lceil \log_2 n \rceil})$ for which t is maximal—in other words, the last such row to have been encoded. This requires $O(n)$ adaptive queries to the NP oracle, each of size $O(n)$. Then output $A(R, i, x)$.

It follows that $\mathsf{PE} \subseteq \mathsf{DTIME}(n^2)^{\mathsf{NP}}$ relative to A, and (by padding) that $\mathsf{PEXP} = \mathsf{P}^{\mathsf{NP}}$. Indeed, once the P^{NP} machine finds the r_ℓ's, it can use them to decide an arbitrary language in $\mathsf{P}^{\mathsf{NP}^{\mathsf{PEXP}}}$, which is why $\mathsf{P}^{\mathsf{NP}} = \mathsf{P}^{\mathsf{NP}^{\mathsf{PEXP}}}$ as well.

For (ii), the change to Theorem 2 is even simpler. Whenever we encode a row $R = (r_1, \ldots, r_\ell)$, instead of setting $A_t(R, i, x) := M_{i,x}(A_{t-1})$ for all i, x, we now set

$$A_t(R, i, x) := M_{i,x}(A_{t-1}) \oplus \bigoplus_{R' \neq R} A_t(R', i, x),$$

where the sum mod 2 ranges over all $R' = (r'_1, \ldots, r'_\ell)$ other than R itself. Then when we are done, by assumption A will satisfy

$$M_{i,x}(A) = \bigoplus_{R=(r_1,\ldots,r_\ell)} A(R, i, x)$$

for all $n \leq 2^\ell$, $i \in \{1, \ldots, n\}$, and $x \in \{0,1\}^n$. So to simulate a PE machine M_i on input x, a $\oplus\mathsf{DTIME}(n)$ machine just needs to return the above sum. Hence $\oplus\mathsf{DTIME}^A(n) = \mathsf{PE}^A$, and $\oplus\mathsf{P}^A = \mathsf{PEXP}^A$ by padding. ■

9. Appendix: Perceptrons

Perceptrons have played an important role in AI and complexity theory since the 1960's [28]. For our purposes, a perceptron is a depth-2 circuit, which consists of a threshold of AND's of negated or non-negated literals. The *size* of the perceptron is the number of AND gates, while the *order* is the maximum fan-in of any AND gate. Suppose a perceptron has size s, and for all $i \in \{1, \ldots, s\}$, let z_i be the output of the i^{th} AND gate. Then the perceptron accepts if and only if

$$c_1 z_1 + \cdots + c_s z_s \geq 0,$$

for some integers $c_1, \ldots, c_s$ called the *weights*. Given a perceptron with size s and weights in $\{-w, \ldots, w\}$, clearly there exists an equivalent perceptron with size ws and weights in $\{-1, 1\}$. For simplicity, from now on we assume that all weights belong to $\{-1, 1\}$.

Our concern here is with a particular problem called ODDMAXBIT. Given an N-bit string $X = x_1 \ldots x_N$, and promised that there exists an i such that $x_i = 1$, let i^* be the maximum such i. Then the ODDMAXBIT problem is to compute $i^* \pmod 2$—that is, to decide whether i^* is odd or even. The idea behind this problem is to model canonical P^{NP}-complete problems [24], such as "Given a Boolean formula φ, does the lexicographically last satisfying assignment to φ end in a 0 or a 1?"

It is not hard to see that ODDMAXBIT can be solved by a perceptron of size $\lceil N/2 \rceil$ and order $\lceil N/2 \rceil$, or by a perceptron of size $2^{N+1} - 1$ and order 1. On the other hand, call a perceptron "small" if it has size $2^{N^{o(1)}}$ and order $N^{o(1)}$. Then Beigel [8] showed the following:

Theorem 13 (Beigel [8]) *No small perceptron can solve ODDMAXBIT.*

Now imagine we have k perceptrons $M_1, \ldots, M_k$, together with k ODDMAXBIT instances $X_1, \ldots, X_k$, each of size N. Also, suppose that each M_j is trying to solve the corresponding instance X_j, but can access bits from *any* of the k instances. Clearly M_j will still be wrong for some values of X_j. But can the perceptrons at least conspire so that they are never all wrong *simultaneously*? Formally, let us say that the "problem set" $X_1, \ldots, X_k$ *defeats* the perceptrons $M_1, \ldots, M_k$, if M_j outputs an incorrect answer to X_j for every $j \in \{1, \ldots, k\}$. Then we can show the following generalization of Theorem 13.

Theorem 14 *For any $k = N^{o(1)}$ small perceptrons, there exists a problem set that defeats them.*

Proof. Follows from simple modifications to the proof of Theorem 2. We can interpret each column of the oracle A as an ODDMAXBIT instance, and each row as an index $i \in \{1, \ldots, N\}$. We can also interpret any $\mathsf{PTIME}^A(T(n))$ machine as a perceptron over the bits of A, with size at most $2^{T(n)}$ and order at most $T(n)$. Let us take A to have k columns and N rows, and let $A(i,j)$ be the bit of A in the i^{th} row and j^{th} column. Also, let $M_1, \ldots, M_k$ be a collection of k perceptrons, each with size at most 2^T and order at most T where $T = N^{o(1)}$. Then $M_j(A)$ is the output of M_j (either 0 or 1) given A.

To create an oracle A that defeats $M_1, \ldots, M_k$, we use the iterative procedure from Theorem 2 (the one for a particular value of n), but with two changes. First, we say that M_j is sensitive to row i, if there *exists* a change to row i that would cause M_j to change its output. To "encode" row i then means to make any such change. Second, we no longer reuse rows from previous iterations, but instead proceed steadily downwards, using a fresh block of $\Theta\left(k^3 T^T\right)$ rows for each iteration. This ensures that when the procedure halts, we obtain a row i^* such that (i) none of the k perceptrons are sensitive to any change to row i^*, and (ii) no row below i^* has yet been modified (i.e. $A(i,j) = 0$ for all $i > i^*$ and all j).

Indeed, we can easily obtain two adjacent rows i^* and $i^* + 1$ that *both* satisfy these properties, with i^* even and $i^* + 1$ odd. We can then defeat $M_1, \ldots, M_k$ as follows: for all $j \in \{1, \ldots, k\}$, set $A(i^*, j) := M_j(A)$ and $A(i^* + 1, j) := 1 - M_j(A)$. This ensures that the j^{th} ODDMAXBIT instance has the answer 1 if M_j outputs 0, and 0 if M_j outputs 1.

All that remains is to show that the procedure halts before running out of rows. Define the polynomial Q as in Theorem 2. One can check that $\deg(Q) = O\left(kT^2\right)$, and that $2^{-O(kT)} \leq Q(A) \leq 2^{O\left(kT^2\right)}$ for all A. It follows that Q can double at most $O\left(kT^2\right)$ times, and hence that there can be at most $O\left(kT^2\right)$ iterations. Also, within each iteration, we want there to exist a perceptron M_j that is sensitive to more than $\deg(Q)^2 = O\left(k^2 T^4\right)$ rows, which means that we want $\Theta\left(k^3 T^4\right)$ rows per iteration. So the total number of rows we need is

$$O\left(kT^2 \cdot k^3 T^4\right) = O\left(k^4 T^6\right) = N^{o(1)},$$

which is less than N for sufficiently large N. ■