On Approximate Majority and Probabilistic Time

Emanuele Viola*

School of Mathematics

Institute for Advanced Study

Princeton, NJ, 08540

`viola@ias.edu`

February 21, 2007

Abstract

We prove new results on the circuit complexity of *Approximate Majority*, which is the problem of computing Majority of a given bit string whose fraction of 1's is bounded away from 1/2 (by a constant). We then apply these results to obtain new relationships between probabilistic time, $\mathrm{BPTime}(t)$, and alternating time, $\Sigma_{O(1)}\mathrm{Time}(t)$. Our main results are the following:

1. We prove that $\left(2^{n^{0.1}}\right)$-size depth-3 circuits for Approximate Majority on n bits have bottom fan-in $\Omega(\log n)$. As a corollary we obtain that $\mathrm{BPTime}(t) \not\subseteq \Sigma_2\mathrm{Time}\left(o(t^2)\right)$ with respect to some oracle. This complements the result that $\mathrm{BPTime}(t) \subseteq \Sigma_2\mathrm{Time}\left(t^2 \cdot \mathrm{poly}\log t\right)$ with respect to every oracle (Sipser and Gács, STOC '83; Lautemann, IPL '83).

2. We prove that Approximate Majority is computable by *uniform* polynomial-size circuits of depth 3. Prior to our work, the only known polynomial-size depth-3 circuits for Approximate Majority were *non-uniform* (Ajtai, Ann. Pure Appl. Logic '83). We also prove that $\mathrm{BPTime}(t) \subseteq \Sigma_3\mathrm{Time}(t \cdot \mathrm{poly}\log t)$. This complements our results in (1).

3. We prove new lower bounds for solving $\mathrm{QSAT}_3 \in \Sigma_3\mathrm{Time}(n \cdot \mathrm{poly}\log n)$ on probabilistic computational models. In particular, we prove that solving QSAT_3 requires time $n^{1+\Omega(1)}$ on Turing machines with a random-access input tape and a sequential-access work tape that is initialized with random bits. No lower bound was previously known on this model (for a function computable in linear space).

*Author supported by NSF grant CCR-0324906. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundations. This research was done while the author was a Ph. D. student at Harvard University, supported by grants NSF CCR-0133096, US-Israel BSF 2002246, and ONR N-00014-04-1-0478; cf. [Vio].

1 Introduction

Understanding the power of probabilistic computation is a central problem in Theoretical Computer Science, and one for which little is known. Essentially, the only non-trivial upper bound that we have on the power of probabilistic computation is the result that probabilistic polynomial time is in the second level of the polynomial-time hierarchy, i.e. BPP $\subseteq \Sigma_2^P$, which was proved in '83 by Sipser and Gács [Sip], and independently by Lautemann [Lau].[1] The results in [Sip, Lau] actually show that probabilistic time $t = t(n)$ with error $1/3$ can be simulated deterministically, using one alternation, with a quadratic blow-up in the running time, i.e. $\mathrm{BPTime}\,(t) \subseteq \Sigma_2\mathrm{Time}\,(t^2 \cdot \mathrm{poly}\log t)$. To the knowledge of the author, there has been no result on whether this quadratic blow-up is necessary. The question of whether the above quadratic blow-up is necessary is closely related to the circuit complexity of *Approximate Majority*, which is the problem of computing Majority of a given bit string whose fraction of 1's is bounded away from $1/2$ (by a constant). In addition to its implications for probabilistic time, the complexity of Approximate Majority is a per se interesting problem which has been widely studied (e.g., [Ajt1, ABO, Sto, Ajt2, CR]). *In this work we prove new results on the circuit complexity of Approximate Majority, and we apply these results to obtain new relationships between probabilistic time and alternating time.*

First, we prove that $\left(2^{n^{0.1}}\right)$-size depth-3 circuits for Approximate Majority on n bits have bottom fan-in $\Omega(\log n)$. As a corollary, we obtain that $\mathrm{BPTime}\,(t) \not\subseteq \Sigma_2\mathrm{Time}\,(o(t^2))$ with respect to some oracle. This shows that the above mentioned quadratic blow-up in the running time of Σ_2 simulations of $\mathrm{BPTime}\,(t)$ is in fact necessary for relativizing techniques (such as those in [Sip, Lau]).

The above result naturally raises the question of whether the quadratic blow-up in the running time of the simulation can be avoided at some higher level of the polynomial-time hierarchy. An involved result by Ajtai [Ajt2] implies that this is indeed possible at *some* level, namely that $\mathrm{BPTime}\,(t) \subseteq \Sigma_c\mathrm{Time}\,(t)$ for some constant c. Ajtai does not bound the constant c, and an analysis of his proof only gives a large constant $c \gg 3$. In this work, we show that there is a quasilinear-time simulation at level $c = 3$, i.e. we show that $\mathrm{BPTime}\,(t) \subseteq \Sigma_3\mathrm{Time}\,(t \cdot \mathrm{poly}\log t)$. Our techniques relativize and thus $c = 3$ is optimal for them, as implied by our oracle result stated in the preceding paragraph. Again, the complexity of Approximate Majority is closely related to our result that $\mathrm{BPTime}\,(t) \subseteq \Sigma_3\mathrm{Time}\,(t \cdot \mathrm{poly}\log t)$. With a slight modification of the techniques used in deriving this result, we also obtain the following: Approximate Majority is computable by *uniform* polynomial-size circuits of depth 3. Prior to our work, the only known polynomial-size depth-3 circuits for Approximate Majority were *non-uniform* [Ajt1]. Tables 1 and 2 summarize the results discussed so far.

The study of the relationship between probabilistic time and alternating time is also motivated by the challenge of proving lower bounds on the running time of probabilistic algorithms for some 'natural' problem. Of course, it is unknown how to prove superlinear time

[1] It is actually known that BPP $\subseteq$ MA $\subseteq S_2^P \subseteq \Sigma_2^P$ [Can, RS]. See [GZ] for discussion of these inclusions. These strengthenings are not directly relevant to our work which, unlike [Can, RS, GZ], focuses on the running time of the simulation.

lower bounds on general computational models (such as multi-tape Turing machines). This is already true for deterministic computation, and probabilistic computation only makes the challenge harder. However, there has been progress in proving lower bounds on restricted models of computation. Much of this progress crucially relies on clever simulations of these restricted models by alternating-time computations. Using our above result that $\mathrm{BPTime}(t) \subseteq \Sigma_3\mathrm{Time}(t \cdot \mathrm{poly}\log t)$, among other ideas, we prove new lower bounds for solving $\mathrm{QSAT}_3 \in \Sigma_3\mathrm{Time}(n \cdot \mathrm{poly}\log n)$ on probabilistic computational models. As it will be apparent from the proofs, our negative result that $\mathrm{BPTime}(t) \not\subseteq \Sigma_2\mathrm{Time}(o(t^2))$ with respect to some oracle shows that substantially different techniques are required to prove similar lower bounds for computing $\mathrm{SAT} = \mathrm{QSAT}_1$ or QSAT_2 (as opposed to QSAT_3 in our results).

We now describe our lower bounds in more detail. The first model we consider is that of probabilistic random-access Turing machines using little space, say at most $n^{.9}$. The works by Beame et al. [BSSV], Allender et al. [AKR$^+$], and Diehl et al. [DvM] prove lower bounds on this model; in particular, Allender et al. [AKR$^+$] and Diehl and van Melkebeek [DvM] prove time lower bounds $t = n^{1+\Omega(1)}$ on this model, but their results hold only for machines with *one-way* access to the random bits. We prove a $t = n^{1+\Omega(1)}$ lower bound for solving $\mathrm{QSAT}_3 \in \Sigma_3\mathrm{Time}(n \cdot \mathrm{poly}\log n)$ on machines with *two-way* (sequential) access to the random bits. To the best of our knowledge, this is the first lower bound of the form $t = n^{1+\Omega(1)}$ on a probabilistic model with random access to the input and two-way access to the random bits. (We elaborate on this model in Section 1.3.)

We then consider the model of Turing machines with two tapes, i.e. the read-only random-access input tape and one sequential-access work tape with no space restrictions. Maass and Schorr [MS] prove a lower bound of the form $t \geq n^{1.22}$ for simulating $\Sigma_1\mathrm{Time}(n)$ on this model. This bound was independently rediscovered in [vMR] where it is also shown that the same bound holds for solving $\mathrm{SAT} \in \Sigma_1\mathrm{Time}(n \cdot \mathrm{poly}\log n)$. While the model in [MS, vMR] is deterministic, we prove that a bound of the form $t = n^{1+\Omega(1)}$ holds, for solving $\mathrm{QSAT}_3 \in \Sigma_3\mathrm{Time}(n \cdot \mathrm{poly}\log n)$, even if the work tape is initialized with random bits and the machine allowed to err with small probability. To the best of our knowledge, no lower bound was previously known on this randomized model.

1.1 Our Results on The Complexity of Approximate Majority

We now describe our results regarding the complexity of *Approximate Majority*. Our main results are summarized and compared to previous work in Table 1.

Approximate Majority is a *promise* problem [ESY] where the task is computing Majority of a given bit string that is promised to have either at least a 2/3 fraction of bits set to 1, or at most a 1/3 fraction of bits set to 1. In this paper we prove the following new lower bound on the *bottom fan-in* of depth-3 (unbounded fan-in) circuits for Approximate Majority, where the bottom fan-in is defined to be the fan-in of the gates adjacent to the input bits.

Theorem 1. *Let C be a depth-3 circuit computing approximate majority on n bits. If the bottom fan-in of C is at most $\log(n)/2$ then the size of C is at least $2^{n^{0.1}}$, for big enough n.*

Table 1: Results on the complexity of computing Approximate Majority on n bits.

Previous Results		
Complexity of Approximate Majority	**Uniformity**	**Reference**
Computable by depth-3 poly(n)-size circuits	non-uniform	[Ajt1]
Computable by depth-$O(1)$ poly(n)-size circuits	Dlogtime-uniform	[Ajt2]
Our Results		
Complexity of Approximate Majority	**Uniformity**	**Reference**
Not computable by depth-3 $2^{n^{0.1}}$-size circuits with bottom fan-in $(\log n)/2$	non-uniform	Th. 1
Computable by depth-3 poly(n)-size circuits	P-uniform	Th. 2

Table 2: Results on simulating probabilistic time by alternating time.

Previous Results		
Inclusion	**Oracle**	**Reference**
$\mathrm{BPTime}\,(t) \subseteq \Sigma_2\mathrm{Time}\,(t^2 \cdot \mathrm{poly}\log t)$	Holds for every oracle	[Sip, Lau]
$\mathrm{BPTime}\,(t) \subseteq \Sigma_{O(1)}\mathrm{Time}\,(t)$	Holds for every oracle	[Ajt2]
Our Results		
Inclusion	**Oracle**	**Reference**
$\mathrm{BPTime}\,(t) \subseteq \Sigma_3\mathrm{Time}\,(t \cdot \mathrm{poly}\log t)$	Holds for every oracle	Th. 3
$\mathrm{BPTime}\,(t) \not\subseteq \Sigma_2\mathrm{Time}\,(o(t^2))$	Holds for some oracle	Th. 5

We point out that in '83 Ajtai [Ajt1] gave a striking probabilistic construction of non-uniform polynomial-size depth-3 circuits for approximate majority.[2] Ajtai's circuits have bottom fan-in $O(\log n)$, which is optimal up to constant factors by Theorem 1. Since Ajtai's construction [Ajt1] is non-uniform, it is natural to ask whether Approximate Majority has uniform polynomial-size depth-3 circuits. We remark that in [Ajt2] Ajtai gives another construction of polynomial-size circuits for Approximate Majority; these circuits are uniform,[3] but they have an unspecified depth $c > 3$. In this paper we show that approximate majority is computable by uniform polynomial-size depth-3 circuits.

Theorem 2. *Approximate majority is computable by* P*-uniform polynomial-size depth-*3 *circuits.*

1.2 Our Results on Simulating Probabilistic Time by Alternating Time

We now describe our results regarding simulating probabilistic time by deterministic alternating time. Our main results are summarized and compared to previous work in Table 2.

On the positive side, we show the following new quasilinear-time simulation of $\mathrm{BPTime}\,(t)$, which holds in any reasonable model of computation that can compute Fourier transforms in time $O(n \cdot \mathrm{poly}\log n)$ (see, e.g., [CLRS]). This simulation was independently obtained by Diehl and van Melkebeek (personal communication, Oct. 2005).

Theorem 3. $\mathrm{BPTime}\,(t) \subseteq \Sigma_3\mathrm{Time}\,(t \cdot \mathrm{poly}\log t)$ *for every constructible function* $t = t(n)$*.*

Since $\mathrm{BPTime}\,(t)$ is closed under complement, we obtain the following corollary, which plays a crucial role in our lower bounds discussed in sections 1.3 and 1.4 below.

Corollary 4. *If* $\Sigma_3\mathrm{Time}\,(n) \subseteq \mathrm{BPTime}\,(n \cdot \mathrm{poly}\log(n))$ *then the quasilinear-time hierarchy collapses to the third level, i.e.* $\bigcup_c \Sigma_c\mathrm{Time}\,(n \cdot \mathrm{poly}\log(n)) = \Sigma_3\mathrm{Time}\,(n \cdot \mathrm{poly}\log(n))$*.*

On the negative side, we prove the following quadratic lower bound on the running time of Σ_2 simulations that *relativize*, i.e. hold with respect to any oracle. We note that all previous simulations [Sip, Lau, Ajt2, Can, RS], as well as ours (Theorem 3), relativize.

Theorem 5. *For every constructible function* $t = t(n)$*,* $\mathrm{BPTime}\,(t) \not\subseteq \Sigma_2\mathrm{Time}\,(o(t^2))$ *with respect to some oracle.*

Theorem 5 shows that, for relativizing techniques, the running time of the Sipser-Gács-Lautemann [Sip, Lau] Σ_2 simulation of $\mathrm{BPTime}\,(t)$ is optimal (up to logarithmic factors), and that consequently the level of our quasilinear-time simulation in Theorem 3 is also optimal, in the sense that it cannot be reduced to 2. For completeness, let us point out that Stockmeyer [Sto] proves that $\mathrm{BPP} \not\subseteq \mathrm{P}^{\mathrm{NP}} \subseteq \Sigma_2^P$ with respect to some oracle. His result is incomparable to our Theorem 5 which addresses the running time of Σ_2 simulations.

[2] While Ajtai [Ajt1] does not explicitly bound the depth of his circuits, it can be verified easily that it equals 3.

[3] Ajtai's construction in [Ajt2] is involved and yields a kind of uniformity, known as Dlogtime, which is stronger than (and in particular implies) polynomial-time uniformity.

1.3 Our Results on Time-Space Lower Bounds

We now discuss our results on time-space lower bounds for probabilistic machines. We prove a new time-space lower bound for simulating $\Sigma_3\mathrm{Time}(n)$ and, in particular, for solving QSAT_3, the problem of deciding the validity of a given Boolean first-order formula with at most 2 quantifier alternations.[4] The computational model on which we prove this negative result is that of a probabilistic Turing machine that can access the tape cells on the input and work tapes by writing a logarithmic-sized index on an associated index tape (random access), while it can only move to adjacent tape cells on the random-bit[5] tape in one time step (sequential access).

In what follows we define our computational model, and then state our lower bound.

Definition 6. *We denote by* $\overleftrightarrow{\mathrm{BP}}\mathrm{TiSp}(t,s)$ *the set of languages accepted by probabilistic Turing machines, with two-sided error, that run simultaneously in time* t *and space* s*, with random access to input and work tapes, and two-way sequential access to the random-bit tape.*

Theorem 7. *For every constant* $\epsilon > 0$, $\Sigma_3\mathrm{Time}(n) \not\subseteq \overleftrightarrow{\mathrm{BP}}\mathrm{TiSp}\left(n^{1+o(1)}, n^{1-\epsilon}\right)$.

Theorem 7 is the first lower bound of the form $t = n^{1+\Omega(1)}$ on a probabilistic computational model with random access to the input tape and *two-way* access to its random bits (for a function computable in, say, linear space). We now elaborate on the strength of models with two-way access to random bits, and then compare our result to the previous ones.

On one-way vs. two-way access to random bits: To appreciate the difference between one-way access and two-way access to random bits, consider log-space computation (L). If one extends L by allowing *one-way* access to random bits, then one gets a complexity class (BPL) that is contained in P. On the other hand, if one allows for *two-way* access to random bits, then one gets a richer complexity class ($\mathrm{BP}\cdot\mathrm{L}$), the power of which is essentially unknown, and conceivably contains NEXP. To further appreciate this difference, we refer to a paper by Nisan [Nis2] which shows that every probabilistic logspace algorithm with one-way access to the random bits can be simulated by a probabilistic logspace algorithm with two-way access to the random bits *with zero error* (i.e., $\mathrm{BPL} \subseteq \mathrm{ZP}\cdot\mathrm{L}$).

An example where two-way access to random bits can be proved to give more power than one-way access is the language of palindromes: it can be recognized in linear time on a sequential one-tape Turing machine with two-way access to the random-bit tape, while it requires time $\Omega(n \cdot \log n)$ if we only allow one-way access to the random-bit tape.[6]

[4] The results for QSAT_3 will not be stated explicitly but follow from techniques in [FLvMV].

[5] It will be always clear from the context whether the word 'random' refers to the machine's ability of addressing tape cells by writing down their index, or to the machine being probabilistic.

[6] The linear-time upper bound can be obtained as follows: On input xy, we accept iff $\langle x, u\rangle = \langle y^R, u\rangle$, where $\langle ., .\rangle$ denotes inner product, y^R the reverse of y, and u the random bits. The lower bound can be derived from the fact that palindromes requires $\Omega(\log n)$ communication for randomized private-coin protocols; see [KN], Example 3.1.

The above discussion begs the question of how large a time bound one can prove on probabilistic computational models with two-way access to the random bits. Our Theorem 7 is a qualitatively new answer to this question.

Comparison with Previous Lower Bounds: Beame et al. [BSSV] prove that there is a function in P that requires time $\Omega(n\sqrt{\log(n/s)/\log\log(n/s)})$ on non-uniform randomized branching programs using space at most s. Since branching programs are more powerful than random-access machines, their lower bound applies to our model (Def. 6), but the time lower bound of $\Omega(n\sqrt{\log n})$ they achieve is weaker than our $n^{1+\Omega(1)}$ bound. Allender et al. [AKR$^+$] prove an $n^{1+\Omega(1)}$ time lower bound on probabilistic random-access machines that have one-way access to random bits and use space at most $n^{1-\epsilon}$, for a function in the counting hierarchy (not believed to be in the polynomial-time hierarchy).[7] In recent and interesting work, Diehl and van Melkebeek [DvM] prove that probabilistic random-access machines that have one-way access to random bits and use space at most n^ϵ require time at least $n^{c-\epsilon}$ to simulate $\Sigma_c\mathrm{Time}(n)$, for every constant $c \geq 2$.[8]

1.4 Our Results on Time Lower Bounds

We now discuss our results about time lower bounds on probabilistic machines. We prove a new lower bound on a probabilistic extension of the two-tape Turing machine model (i.e. a Turing machine with a read-only input tape and one sequential-access work tape with no space restrictions). Our probabilistic extension, denoted $\mathrm{BPTime}_1(t)$, is obtained by initializing the work tape of the machine with random bits, and allowing the machine to err with small probability.

Theorem 8. $\Sigma_3\mathrm{Time}(n) \not\subseteq \mathrm{BPTime}_1\left(n^{1+o(1)}\right)$.

Theorem 8 is the first lower bound on the model $\mathrm{BPTime}_1(t)$ (for a function computable in, say, linear space). In fact, our lower bound applies to a single model that simultaneously extends $\mathrm{BPTime}_1\left(n^{1+o(1)}\right)$ and the previously considered $\overleftrightarrow{\mathrm{BP}}\mathrm{TiSp}\left(n^{1+o(1)}, n^{1-\epsilon}\right)$ (Def. 6); see the full version of this paper.

1.5 Organization

This paper is organized as follows. In Section 2 we prove our lower bound for approximate majority (Theorem 1), and informally explain why it implies our oracle separation in Theorem 5. In Section 3 we prove that $\mathrm{BPTime}(t) \subseteq \Sigma_3\mathrm{Time}\left(t \cdot \log^2 t\right)$ (Theorem 3). In Section 4 we prove that Approximate Majority is computable by uniform polynomial-size circuits of

[7] In fact, the results in [AKR$^+$] apply to the more general setting of *unbounded* error probabilistic computation.

[8] In [DvM] they also point out that their space bound becomes $n^{.25-\epsilon}$ on machines running in time $n^{1+\epsilon}$. This space bound (for the particular case of machines running in time $n^{1+\epsilon}$) later has been improved to $n^{.5-\epsilon}$ (for simulating $\Sigma_2\mathrm{Time}(n)$) and to $n^{1-\epsilon}$ (for simulating $\Sigma_3\mathrm{Time}(n)$). These improvements have been obtained by Diehl and van Melkebeek (personal communication, Oct. 2005), and independently by us.

depth 3 (i.e., Theorem 2). In Section 5 we prove our time-space lower bound (Theorem 7). In Section 6 we mention a few open problems.

Appendix A contains the proof of our time lower bound on Turing machines (Theorem 8). Appendix B discusses the dependence of our results on the error probability ϵ of the BPTime (t) machines, while we set $\epsilon = 1/3$ in the rest of the paper. In Appendix C we prove why our lower bound for approximate majority (Theorem 1) implies our oracle separation in Theorem 5.

2 Lower Bound for Approximate Majority

In this section we prove our lower bound on the bottom fan-in of (unbounded fan-in) depth-3 circuits computing approximate majority. At the end of the section we compare our techniques to previous ones, and informally discuss why our lower bound implies our oracle separation in Theorem 5.

Let us begin by formally defining approximate majority and restating our main result.

Definition 9. Approximate Majority *is the following promise problem:*

$$\begin{aligned} \text{ApprMaj}_{YES} &:= \{x : \text{ at least } 2|x|/3 \text{ bits of } x \text{ are set to } 1\}, \\ \text{ApprMaj}_{NO} &:= \{x : \text{ at most } |x|/3 \text{ bits of } x \text{ are set to } 1\}. \end{aligned}$$

Theorem (1, restated)**.** *Let C be a depth-3 circuit computing approximate majority on n bits. If the bottom fan-in of C is at most $\log(n)/2$ then the size of C is at least $2^{n^{0.1}}$, for big enough n.*

We now explain the proof of Theorem 1. It is convenient to work with the following distribution, which generates $x \in \text{ApprMaj}_{NO}$ with sufficiently high probability (for our purposes).

Definition 10. *Let D^n be the distribution on $\{0,1\}^n$ that sets each bit to 1 independently with probability $1/3$ (and to 0 with probability $2/3$).*

The core of the proof of Theorem 1 is the following lemma.

Lemma 11. *Let φ be a DNF on n variables with terms of size at most k. If $\varphi(x) = 1$ for every $x \in \{0,1\}^n$ with at least $2n/3$ bits set to 1, then $\Pr_{x \in D^n}[\varphi(x) = 0] \le e^{-\frac{n}{3 \cdot k^2 \cdot 3^k}}$.*

Before we explain the intuition for the proof of Lemma 11, let us see why it implies Theorem 1.

Proof of Theorem 1 assuming Lemma 11. First note that $\Pr_{x \in D^n}[x \in \text{ApprMaj}_{NO}] \ge 1/3$ by the standard Central Limit Theorem,[9] and therefore we have that $\Pr_{x \in D^n}[C(x) = 0] \ge 1/3$.

Assume that the output gate of C is an AND gate (otherwise we can negate the circuit and carry through essentially the same argument as below for approximate majority with

[9] Alternatively, one can change the parameter $1/3$ in the distribution D^n to any other smaller constant, and prove, using Markov inequality, a bound which is enough to obtain our results modulo different constants.

YES and NO swapped). So $C = \bigwedge_i \varphi_i$, where each φ_i is a OR-AND depth-2 circuit, i.e. a DNF. Note that, by definition of AND, for every $x \in \text{ApprMaj}_{YES}$ we have $\varphi_i(x) = 1$ for all i, while for random $x \in D^n$ we have that with probability at least 1/3 there is an i such that $\varphi_i(x) = 0$. By an averaging argument we can fix a DNF $\varphi = \varphi_i$ such that: (1) for every $x \in \text{ApprMaj}_{YES}$ we have $\varphi(x) = 1$; (2) $\Pr_{x\in D^n}[\varphi(x) = 0] \geq 1/(3 \cdot |C|)$, where $|C|$ is the size of the circuit C, i.e. the number of its gates. Note that if C has bottom fan-in at most k then the same holds for the subcircuit φ fixed above, i.e. φ is a DNF with term size at most k. By Lemma 11, we have

$$\frac{1}{3 \cdot |C|} \leq e^{-n/(3\cdot k^2 \cdot 3^k)} = e^{-n^{\Omega(1)}},$$

and so we conclude $|C| \geq e^{n^{\Omega(1)}}$. In fact, $|C| \geq 2^{n^{0.1}}$, because $3^k = n^{(\log_2 3)/2} \approx n^{.792}$. □

Overview of the Proof of Lemma 11: The proof of Lemma 11 is an inductive argument inspired by a recent switching lemma by Segerlind et al. [SBI], which we discuss later in this section. A similar argument is also a component of a technically intricate lower bound on the round complexity of protocols for two-party random selection [SV].

Let us define a *covering* of a DNF φ as a subset Γ of the variables such that each term in φ contains at least one variable, possibly negated, in Γ. For example, the smallest covering of the DNF $\varphi(x_1, x_2, x_3, x_4) := (x_1x_2x_3)\text{OR}(\neg x_1)\text{OR}(x_2x_4)$ is $\Gamma := \{x_1, x_2\}$. To prove Lemma 11 we then argue as follows. Consider the smallest covering Γ of φ. There are two cases: Either $|\Gamma| > n/(3k)$ or $|\Gamma| \leq n/(3k)$. (Recall $k = (\log n)/2$ is the maximum term size of φ.)

Case $|\Gamma| > n/(3k)$: In this case φ must contain at least $|\Gamma|/k = n/\text{poly}\log(n)$ *disjoint* terms, where disjoint means that no two terms share a variable. Such terms can be found greedily, using the fact that the terms are of size at most k. Now, the probability that $\varphi(x) = 0$ for random $x \in D^n$ is at most the probability that all these $n/\text{poly}\log(n)$ terms evaluate to 0. Since the terms are disjoint, this can be bound by raising to the power of $n/\text{poly}\log(n)$ the probability that a single term is 0. But since terms have at most $k = (\log n)/2$ variables, the probability that a term is 0 can be shown to be at most $(1-1/n^\epsilon)$, for a constant $\epsilon < 1$. Thus we have

$$\Pr_{x\in D^n}[\varphi(x) = 0] \leq \left(1 - \frac{1}{n^\epsilon}\right)^{n/\text{poly}\log(n)} = 2^{-n^{\Omega(1)}},$$

which proves Lemma 11.

Case $|\Gamma| \leq n/(3k)$: In this case by an averaging argument we fix the variables in Γ and obtain a new DNF φ' such that $\Pr_{x\in D^n}[\varphi'(x) = 0] \geq \Pr_{x\in D^n}[\varphi(x) = 0]$. Then we iterate the argument on φ' (i.e. we consider the size of the smallest covering of φ', etc.).

We keep iterating the argument until we obtain a DNF φ' whose smallest covering has size at least $n/(3k)$, or a DNF φ' that computes a constant function. By construction, $\Pr_{x\in D^n}[\varphi'(x) = 0] \geq \Pr_{x\in D^n}[\varphi(x) = 0]$, so we only need to worry about the case $\varphi' \equiv 0$ (otherwise Lemma 11 is proved as stated above). We rule out the case $\varphi' \equiv 0$ by exhibiting $x \in \text{ApprMaj}_{YES}$ such that $\varphi(x) = 0$, which contradicts the hypothesis of Lemma 11. To construct such an x, note that each iteration assigns values to the variables in a covering of the DNF, and so at each iteration the term size of the DNF decreases (since each term has at least one variable in the covering). Therefore we iterate the argument at most k times. Since each iteration fixes at most $n/(3k)$ variables, in the end we have fixed at most $k \cdot n/(3k) = n/3$ variables. By setting all the remaining variables to 1 we set at least $2n/3$ variables to 1 and thus we have $x \in \text{ApprMaj}_{YES}$ such that $\varphi(x) = \varphi'(x) = 0$, which contradicts the hypothesis of Lemma 11 and concludes this proof sketch.

We now present the formal proof of Lemma 11.

Proof of Lemma 11. We define a *covering* Γ of a DNF φ to be a subset of variables such that each term in φ contains at least one variable in Γ (possibly negated). Consider the following procedure:

Procedure(φ)
If φ computes a constant function then *stop*. Remove all terms in φ that compute the constant function 0 (e.g. $x_1 \wedge \neg x_1$). Let Γ be a minimum-size covering of the terms of φ. If $\lvert\Gamma\rvert \geq n/(3 \cdot k)$ then *stop*. Partition the v variables of $\varphi(x)$ into $x = y \bigcup z$, where $y = \Gamma$ and $z \bigcap \Gamma = \emptyset$. Fix an assignment $y = a$ such that $\Pr_{z \in D^{v-\lvert\Gamma\rvert}}[\varphi(a \cdot z) = 0] \geq \Pr_{x \in D^v}[\varphi(x) = 0]$. (Such an assignment exists by an averaging argument.) Consider the new DNF $\varphi'(z) := \varphi(a \cdot z)$ obtained by hardwiring a in φ. Repeat the procedure on the DNF φ'.

Claim 12. *The procedure stops after at most k iterations.*

Proof. At each iteration we fix the variables y in a covering Γ of φ. Since by definition of covering each term of φ contains a variable from Γ, this decreases the maximum term size of φ. When the term size is 0 then the DNF is a constant and we stop. ☐

The procedure constructs a sequence of DNFs $\varphi = \varphi_1, \varphi_2, \ldots, \varphi_t$, where DNF φ_i is on n_i variables. We have $t \leq k$ by the above claim. Also, by construction we have

$$\Pr_{x \in D^n}[\varphi(x) = 0] = \Pr_{x \in D^{n_1}}[\varphi_1(x) = 0] \leq \Pr_{x \in D^{n_2}}[\varphi_2(x) = 0] \leq \ldots \leq \Pr_{x \in D^{n_t}}[\varphi_t(x) = 0]. \quad (1)$$

To finish the proof, we simply analyze what happens when the procedure stops. If the procedure stops at the i-th iteration because φ_i computes a constant function, we argue as follows: If $\varphi_i(x) = 1$ for every x then $\Pr_{x \in D^{n_i}}[\varphi_i(x) = 0] = 0$, and by Equation (1) we have $\Pr_{x \in D^n}[\varphi(x) = 0] \leq 0$, which proves the lemma.

If $\varphi_i(x) = 0$ for every x then notice that at each iteration we fix at most $n/(3k)$ variables, and therefore by Claim 12 φ_i equals φ with at most $k \cdot n/(3k) = n/3$ variables fixed. By setting all the remaining variables to 1, we have found an input $\hat{x}$ with at least $2n/3$ bits set to 1 such that $\varphi(\hat{x}) = 0$, which contradicts the hypothesis of the lemma.

Otherwise, the procedure stops at the i-th iteration because every covering of φ_i has size at least $n/(3k)$. Therefore there is a set S of at least $n/(3k^2)$ disjoint terms, where disjoint means that no two terms share a variable. To see why this is true, let S be a *maximal* set of disjoint terms. The union of the variables in the terms in S is a covering of φ_i of size at most $|S| \cdot k$: if it were not a covering, we could add a term to S, contradicting its maximality. Thus $|S| \geq (n/3k)/k$.

Then we can bound $\Pr_x[\varphi(x) = 0]$ as follows:

$$\Pr_{x \in D^n}[\varphi(x) = 0] \leq \Pr_{x \in D^{n_i}}[\varphi_i(x) = 0] \leq \left(1 - \frac{1}{3^k}\right)^{n/(3k^2)} \leq e^{-\frac{n}{3 \cdot k^2 \cdot 3^k}},$$

where the first inequality follows by Equation (1), and the second follows by considering the terms on disjoint sets of variables in φ_i. Specifically, we notice that the probability that any of them is 0 is at most $(1 - 1/3^k)$. This holds because in the procedure we always remove terms computing the constant function 0, and thus each term must be 1 under at least one assignment of its (at most k) variables. This assignment is picked under the distribution D^{n_i} with probability at least $1/3^k$. Moreover, these events are independent for the $n/(3 \cdot k^2)$ terms with disjoint variables. $\square$

Our proof vs. previous techniques: A standard approach to prove lower bounds for constant-depth circuits is to use a *switching lemma* (see, e.g., [Hås]). Håstad's switching lemma [Hås] cannot be used directly to prove lower bounds for the promise problem Approximate Majority. This is because to apply this lemma to a circuit with bottom fan-in k, we need to assign values to at least a $(1 - 1/k)$ fraction of the variables. The switching lemma would assign 0 to roughly half of this fraction of variables, and so as soon as $k \geq 3$, we would produce an input $x \notin \mathrm{ApprMaj}_{YES} \bigcup \mathrm{ApprMaj}_{NO}$.

Recently, Segerlind et al. [SBI] proved a new switching lemma that assigns values to much fewer variables than does Håstad's switching lemma. One can apply this switching lemma [SBI] to prove that small depth-3 circuits for approximate majority on n bits require bottom fan-in at least $\Omega(\sqrt{\log n})$; however, we were unable to apply their results to circuits with bigger bottom fan-in, such as $(\log n)/2$.

We have recently found out that Razborov [Raz] improved the parameters of the above switching lemma by Segerlind et al. [SBI]. Using Razborov's switching lemma and making a few observations, one can prove our lower bound (Theorem 1) (up to different constants). Our proof is arguably simpler than a proof based on the switching lemma.

The connection between Approximate Majority and simulating $\mathrm{BPTime}(t)$**:** For completeness, we now review why our circuit lower bound for approximate majority in Theorem 1 implies our oracle separation in Theorem 5. A more formal proof is given in Appendix C.

Consider simulating a probabilistic machine $M \in \mathrm{BPTime}(t)$ by a machine in $\Sigma_2\mathrm{Time}(t')$. Given an input x, by definition of $\mathrm{BPTime}(t)$ we are *promised* that either $\Pr_u[M(x; u) = 1] \geq 2/3$ or $\Pr_u[M(x; u) = 1] \leq 1/3$. This corresponds to an approximate majority instance Y of exponential length $|Y| = 2^{|u|} = 2^t$, where the i-th bit of Y is $M(x; i)$ (the equality

$2^{|u|} = 2^t$ holds because the machine runs time t). Thus, intuitively, the task of the Σ_2 machine is to distinguish $Y \in \text{ApprMaj}_{YES}$ from $Y \in \text{ApprMaj}_{NO}$. It is well known from [FSS] that the Σ_2 computation can be seen as an unbounded fan-in circuit of depth 3: the two quantifiers give rise to the first two levels of the circuit (OR-AND). After its quantifications, the Σ_2 simulation is followed by a deterministic computation running in time t'. Since computing $M(x; u)$ for fixed $(x; u)$ takes (deterministic) time t, the Σ_2 computation can depend on at most $k := t'/t$ evaluations of M. We can write this part of the Σ_2 computation as a depth-2 (AND-OR) circuit with bottom fan-in k. Finally, by collapsing the top AND gate of this circuit with the AND arising from the second quantifier, we obtain a small circuit of depth 3 with bottom fan-in $k = t'/t$. By our lower bound (Theorem 1), small depth-3 circuit for Approximate Majority on $|Y|$ bits have bottom fan-in $\Omega(\log |Y|) = t$, and therefore we obtain that $t' \geq \Omega(t^2)$.

The above sketch can be shown to be exact for relativizing simulations, using the standard convention that the oracle tape is erased after each query (see, e.g., [BDG]). We stress that this convention is natural: it is intuitively capturing the fact that the $\Sigma_2\text{Time}(t')$ machine cannot run the code of the $\text{BPTime}(t)$ machine more than t'/t times, since each execution of the latter runs in time t. Finally, we would like to point out that our result in Theorem 5 also applies to simulations of $\text{BPTime}(t)$ by $\Sigma_2\text{Time}(t')$ that are *black-box* (as opposed to relativizing), because black-box simulations relativize (folklore).

3 $\text{BPTime}(t) \subseteq \Sigma_3\text{Time}(t \cdot \text{poly}\log t)$

In this section we discuss the main ideas behind the proof of our result that $\text{BPTime}(t) \subseteq \Sigma_3\text{Time}(t \cdot \text{poly}\log t)$, i.e. Theorem 3. These ideas are also a component of our proof that Approximate Majority has uniform polynomial-size circuits of depth-3, i.e. Theorem 2, though more work is needed for that (see Section 4).

Let us focus on the case $t = n$ and review Lautemann's proof of $\text{BPTime}(n) \subseteq \Sigma_2\text{Time}(n^2 \cdot \text{poly}\log n)$, as this is the starting point of our argument. Let $M(x; u)$ be a probabilistic machine using n random bits u. Lautemann's approach is to *guess* n 'shifts' $w_1 \in \{0,1\}^n, w_2 \in \{0,1\}^n, \ldots, w_n \in \{0,1\}^n$ and check if *for every* $u \in \{0,1\}^n$ we have

$$\Big(M(x; u \oplus w_1) = 1\Big) \bigvee \Big(M(x; u \oplus w_2) = 1\Big) \bigvee \ldots \bigvee \Big(M(x; u \oplus w_n) = 1\Big). \qquad (2)$$

Noting that we use two quantifiers, each ranging over at most n^2 bits, and that the computation in Equation 2 takes time n^2, we have that this is a $\Sigma_2\text{Time}(n^2)$ simulation. The proof of correctness is a counting argument, which we omit.[10]

There are two reasons why this simulation takes at least quadratic time. The first is that the computation in Equation (2) runs M for n times (over fixed random bits). Since M

[10]Actually, this approach requires that the error probability of M is at most $1/n^2$. We can achieve this by paying an extra $O(\log n)$ factor in the running time, if M starts off with error $1/3$. We ignore this issue to simplify the exposition.

runs in time $O(n)$, this takes at least time n^2. The second reason is that we initially guess $n^2 = |w_1, w_2, \ldots, w_n|$ bits.

Let us focus on the first problem, that is, the fact that computing Equation (2) requires quadratic time. This problem *cannot* be avoided using relativizing techniques: Our negative result (Theorem 5) shows that every relativizing Σ_2 simulation *must* run M at least $\Omega(n)$ times, and thus must have total run time at least n^2. As a first step towards our quasilinear Σ_3 simulation, we observe that the computation in Equation (2) is an OR over n evaluations of M; thus, we can use another quantifier for this OR, and then run M once.

To obtain a Σ_3 simulation that runs in quasilinear time, we still have to solve the second problem, the quantification over $w_1, w_2, \ldots, w_n$. As it turns out, the only property of these w_i's that is used in Lautemann's proof is a *hitting* property, i.e. for any 'big' set $A \subseteq \{0,1\}^n$, the probability over random $w_1, w_2, \ldots, w_n$ that none of the w_i's lands in A is exponentially small in n. It is well known that there are more 'randomness-efficient' ways to generate w_i's with this property (see, e.g., [Gol]). In particular, it is possible to generate such w_i's using a *hitting* generator with a seed of length $|\sigma| = O(n)$. We use this approach: instead of guessing n^2 bits for $w_1, w_2, \ldots, w_n$, we only guess $O(n)$ bits σ and let $w_i := G(\sigma)_i$, where $G(\sigma) = G(\sigma)_1 \cdot G(\sigma)_2 \cdots G(\sigma)_n \in (\{0,1\}^n)^n$ for an appropriate generator G.

However, for our simulation we need a generator that runs in quasilinear time in the sense that given σ and i we need to compute the i-th output $G(\sigma)_i = w_i \in \{0,1\}^n$ of the generator in quasilinear time. Well-known generators based on *random walks on expander graphs* achieve seed length $O(n)$ (see, e.g., [Gol]), but we do not know how to compute any of them in less than quadratic time.[11]

Instead, we use a generator by Nisan [Nis1] which has slightly worse seed length $|\sigma| = O(n \cdot \log n)$, but which on the other hand can be computed in time $O(n \cdot \text{poly}\log n)$: to compute one output of the generator we only have to evaluate $\log n$ *pairwise independent hash functions* $h : \{0,1\}^n \to \{0,1\}^n$. Using hash functions based on convolution or finite field arithmetic, one can compute each such hash function in time $O(n \cdot \text{poly}\log n)$ using the Fast Fourier Transform.

Remark 13 (On the existence of simpler proofs of the fact that $\text{BPTime}\,(n) \subseteq \Sigma_3\text{Time}\,(n \cdot \text{poly}\log n)$)**.** *As explained above, our proof of the fact that* $\text{BPTime}\,(n) \subseteq \Sigma_3\text{Time}\,(n \cdot \text{poly}\log n)$ *(Theorem 3) is based on Lautemann's proof that* $\text{BPTime}\,(n) \subseteq \Sigma_2\text{Time}\,(n^2 \cdot \text{poly}\log n)$. *Some researchers have suggested that a simpler proof of Theorem 3 can be obtained following the simplification of Lautemann's proof in [GZ, Section 3.2]. We observe that such a simpler proof seems to require computing walks on expander graphs in quasilinear time, a non-trivial fact that was proved concurrently with this work by Diehl and van Melkebeek (personal communication, Oct. 2005). On the other hand, our proof relies on the standard fact that Fourier transform can be implemented in quasilinear time.*

Let us restate the main result of this section, and then formally prove it.

[11] Concurrently with our work, Diehl and van Melkebeek show how to compute walks on the Margulis-Gabber-Galil expander graph [Mar, GG, JM] in quasilinear time (personal communication, Oct. 2005).

Theorem (3, restated)**.** BPTime $(t) \subseteq \Sigma_3$Time $(t \cdot \text{poly}\log t)$ *for every constructible function* $t = t(n)$.

The proof of Theorem 3 uses as a component a hitting generator by Nisan.

Lemma 14 ([Nis1], Theorem 3)**.** *For every r and $k \leq 2^r$, there exists a generator*

$$N_k : \{0,1\}^l \to (\{0,1\}^r)^k, \ \ N_k(\sigma) = N_k(\sigma)_1 \cdot N_k(\sigma)_2 \cdots N_k(\sigma)_k,$$

such that:

1. *N_k has seed length $l = |\sigma| = O(r \log k)$,*
2. *for every set $A \subseteq \{0,1\}^r$ we have*

$$\left| \Pr_\sigma[\forall i \leq k : N_k(\sigma)_i \in A] - \left(\frac{|A|}{2^r}\right)^k \right| \leq 4^{-r},$$

3. *given a seed σ and $i \leq k$ we can compute $N_k(\sigma)_i \in \{0,1\}^r$ in time $(r \cdot \text{poly}\log r)\log k$.*

Proof. Items (1) and (2) in Lemma 14 are proved in Nisan's paper [Nis1]. To prove Item (3), we note that computing one r-bit output of the generator requires computing $\log k$ pairwise independent hash functions. Using hash functions based on convolution or finite field arithmetic one can compute each such hash function in time $O(r \cdot \text{poly}\log r)$ using the Fast Fourier Transform (see, e.g., [Nis1] Section 2.2 for the definition of hash functions based on convolution, and Theorem 30.8 in [CLRS] for the use of the Fast Fourier Transform to compute convolution). □

Proof of Theorem 3. We prove BPTime $(n) \subseteq \Sigma_3$Time $(n \cdot \text{poly}\log n)$, the inclusion for generic time bound $t = t(n)$ then follows by a standard padding argument (see, e.g., [Pap]). Let M be an algorithm in BPTime (n) and let us write $M(x; u)$ for algorithm M on input x and random bits u.

Claim 15. *Let M be an algorithm in* BPTime (n). *There is an algorithm M' that accepts the same language as M such that: M' has error probability at most $1/n^2$, M' uses $O(n)$ random bits, and given x, u we can compute $M'(x; u)$ in time $O(n \cdot \log n)$.*

We postpone the proof of Claim 15 and we proceed with the proof of the theorem. (Note that if one is willing to have M' use $O(n \log n)$ random bits, instead of $O(n)$, then one can obtain such a M' by simply taking the majority of $O(\log n)$ repetitions of $M(x)$ with independent random bits. Using this instead of Claim 15 and proceeding with the proof gives a simulation with worse running time, but still $n \cdot \text{poly}\log n$. However, the better parameters achieved here are used later in Section 4.)

Let M' be the algorithm from Claim 15 and let $r = r(n) = O(n)$ be the number of random bits used by M' on input of length n. Let $N_k : \{0,1\}^l \to (\{0,1\}^r)^k$ be the generator from Lemma 14 with $k = r$. Note that the seed length of N_k is $|\sigma| = l = O(r \log k) = O(n \cdot \log n)$.

Now consider the Σ_3 machine that accepts input x if and only if

$$\exists \sigma \in \{0,1\}^l \, \forall u \in \{0,1\}^r \, \exists i \leq k : M'\left(x; N_k(\sigma)_i \oplus u\right) = 1,$$

where $\oplus$ denotes bitwise xor.

Correctness: Assume $M'(x) = 1$. We must show that the Σ_3 machine accepts. Consider

$$\Pr_\sigma[\exists u \in \{0,1\}^r \, \forall i \leq k : M'\left(x; N_k(\sigma)_i \oplus u\right) = 0]. \tag{3}$$

We show that this probability is less than 1 and therefore that the machine accepts. By a union bound this probability (3) is at most

$$2^r \cdot \Pr_\sigma[\forall i \leq k : M'\left(x; N_k(\sigma)_i\right) = 0].$$

(Note we removed $\oplus u$ because this just shifts the space of random bits of the algorithm and does not change the probability.) Since $M'(x) = 1$, and M' has error at most $1/n^2$, we get by Lemma 14 that the probability is at most (recall $k = r$)

$$2^r \left((1/n^2)^k + 4^{-r}\right) < 1.$$

Now assume that $M'(x) = 0$. Fix any seed σ, we must show that $\exists u \in \{0,1\}^r \, \forall i \leq k : M'\left(x; N_k(\sigma)_i \oplus u\right) = 0$, and thus the machine rejects. Consider

$$\Pr_u[\exists i \leq k : M'\left(x; N_k(\sigma)_i \oplus u\right) = 1.$$

Again, we show that this probability is less than 1 and therefore that the machine accepts. By a union bound this probability is at most

$$k \cdot \Pr_u[M'\left(x; u\right) = 1].$$

(Note again we removed $N_k(\sigma)_i \oplus$ because this just shifts the space of random bits of the algorithm and does not change the following analysis.) Since M' has error at most $1/n^2$ this probability is at most $k/n^2 = r/n^2 = O(n \cdot \log n)/n^2 < 1$.

Complexity: The machine uses three quantifiers, each on at most $O(n \cdot \log n)$ bits, by inspection. Each computation branch only runs $M'\left(x; N_k(\sigma)_i \oplus u\right)$ once. $N_k(\sigma)_i$ can be computed in time $(r \cdot \operatorname{poly}\log r) \cdot \log k = n \cdot \operatorname{poly}\log n$ by Lemma 14, and for given x, u', $M'(x; u')$ can be computed in time $O(n \cdot \log n)$ by Claim 15. □

Proof sketch of Claim 15. We use the $O(n)$ random bits of M' to encode a walk $w_1, w_2, \ldots, w_l$ of length $l = O(\log n)$ on an expander graph of $2^{O(n)}$ vertices. We then define $M'(x; u) := \mathrm{Maj}_{j \leq l} M(x; w_j)$. The bound on the error probability of the algorithm follows by the Chernoff Bound for random walks on expander graphs [Gil]. Using the expander graph in [Mar] (the expansion of which is analyzed in [GG, JM]), we can compute all the w_i's in time $O(n \cdot \log n)$: computing w_i from w_{i-1} amounts to a constant number of additions, which can be done in time $O(n)$. □

4 Uniform Depth-3 Circuits for Approximate Majority

In this section we prove the following theorem regarding the complexity of computing approximate majority.

Theorem (2, restated)**.** *Approximate majority is computable by* P*-uniform polynomial-size depth-*3 *circuits.*

The proof of Theorem 2 makes use of the following hitting generator (see Section 3 for a discussion of hitting generators).

Lemma 16. *For every n there exists a polynomial-time computable generator*

$$Z : \{0,1\}^{O(n)} \to (\{0,1\}^n)^k\,,\ \ Z(\sigma) = Z(\sigma)_1 \cdot Z(\sigma)_2 \cdots Z(\sigma)_k,$$

with $k = O(n/\log n)$, such that for every set $A \subseteq \{0,1\}^n$ of size $|A| \leq 2^n/n^2$ we have

$$\Pr_\sigma[\forall i \leq k : Z(\sigma)_i \in A] \leq 4^{-n}.$$

Proof sketch of Lemma 16. Use the input σ to encode a random walk (started at a random vertex) of length $c \cdot n/\log n$ on an $n^{O(1)}$-regular expander graph on 2^n vertices with second largest eigenvalue $\lambda = 1/n^2$, for a constant c to be determined later.[12] Note that one can encode such a walk using $(c \cdot n/\log n) \cdot \log n^{O(1)} = O(n)$ bits. Kahale [Kah] shows that the probability that all the t steps of the random walk fall inside a fixed set of density $\mu := 1/n^2$ can be bounded as

$$\mu \cdot (\mu + (1-\mu) \cdot \lambda)^{t-1} \leq (1/n^{\Omega(1)})^{c \cdot n/\log n} \leq 4^{-n},$$

where the last inequality holds by choosing a sufficiently large constant c. □

We now proceed with the proof of Theorem 2.

Proof of Theorem 2. The main ideas of the proof are the same as those of the proof of Theorem 3. It is convenient to think of an approximate majority instance of length N as $M(0) \cdot M(1) \cdots M(N)$, where $M(i)$ is the i-th bit of the instance.

Let $Z : \{0,1\}^{O(n)} \to (\{0,1\}^n)^k$ be the hitting generator from Lemma 16, where $k = O(n/\log n)$. We decide approximate majority by checking whether the following equation is true:

$$\exists \sigma \in \{0,1\}^{O(n)}\, \forall u \in \{0,1\}^n\, \exists i \leq k : M'(Z(\sigma)_i \oplus u) = 1, \tag{4}$$

where $\oplus$ denotes bitwise xor, and M' denotes the machine with error $1/n^2$ from Claim 15, which we think of as an approximate majority instance with amplified gap (i.e., either at least a $1 - 1/n^2$ fraction of input bits is set to 1, or at most a $1/n^2$ fraction of input bits is set to 1).

[12]Such an expander G can be obtained by taking a λ-biased set $S \subseteq \{0,1\}^n$ of size poly(n) [NN, AGHP] and setting $G := (\{0,1\}^n, \{(x,y) : x - y \in S\})$. Alternatively, we can take a $O(\log n)$ power of a constant-degree expander.

Correctness: The proof of correctness is the same as that of Theorem 3.

Computable by uniform $\mathrm{poly}(N)$*-size circuits of depth* 3*:* We note that the computation of $M'(G(\sigma)_i \oplus u)$ is the majority of $O(\log n)$ evaluations of M, and therefore the computation of

$$\exists i \leq k : M'(G(\sigma)_i \oplus u) = 1$$

only depends on $k \cdot O(\log n) = O(n)$ evaluations of M. We write this computation as a CNF of size $2^{O(n)} = \mathrm{poly}(N)$. We then collapse the output AND of this CNF with the second quantifier in Equation (4). Finally, since the first two quantifiers in Equation (4) range over $O(n)$ bits, they give rise to gates with fan-in $2^{O(n)} = \mathrm{poly}(N)$, and thus the whole computation in Equation (4) can be written as a $\mathrm{poly}(N)$-size circuit of depth 3.

It is easy to check that the circuit can be constructed in time $\mathrm{poly}(N)$. (Note that the generator Z in Lemma 16 is computable in time $\mathrm{poly}(n) = \mathrm{poly}\log N$.) □

On Depth-3 Circuits vs. Σ_3 time: The above proof of Theorem 2 is similar to the proof of Theorem 3 (which is the result that $\mathrm{BPTime}(t) \subseteq \Sigma_3\mathrm{Time}(t \cdot \mathrm{poly}\log t)$), and thus it is worth pointing out why the two results (Theorem 2 and Theorem 3) actually are incomparable.

The depth-3 circuit in the above proof of Theorem 2 is P-uniform, i.e. it can be constructed in time polynomial in its size. P-uniformity is too loose to obtain Theorem 3 for which one needs the circuit to satisfy the following stronger uniformity condition: given the indices to two gates in the circuit, it is possible to decide whether the two gates are connected in time quasilinear in the length of their indices. It is conceivable that the circuit does satisfy this stronger uniformity condition, but this does not seem straightforward to us.

Conversely, the circuit obtained in Theorem 3 (i.e. $\mathrm{BPTime}(t) \subseteq \Sigma_3\mathrm{Time}(t \cdot \mathrm{poly}\log t)$) does satisfy the above stronger uniformity condition, but has superpolynomial size and depth 4. (The circuit has depth 4 because the 3 alternations are followed by a computation that depends on several evaluations of the $\mathrm{BPTime}(t)$ machine. This computation gives rise to another layer of gates in the circuit.)

5 Time-Space Lower Bound

In this section we prove our time-space lower bound (Theorem 7). We start with an informal overview of the techniques, and then proceed with a formal proof sketch.

Overview of Techniques: Our time-space lower bound for $\Sigma_3\mathrm{Time}(n)$ is inspired by an interesting recent paper by Diehl and van Melkebeek [DvM] which, among other results, proves that $\Sigma_3\mathrm{Time}(n) \not\subseteq \overrightarrow{\mathrm{BP}}\mathrm{TiSp}(n^{1+\epsilon}, n^{\epsilon})$ for some constant ϵ, where $\overrightarrow{\mathrm{BP}}\mathrm{TiSp}(t, s)$ denotes the class of problems that can be solved simultaneously in time t and space s on a probabilistic random-access Turing machine with one-way access to its random bits. It is convenient for this exposition to think of the approach in [DvM] as giving a sublinear-time Σ_3 simulation of $\overrightarrow{\mathrm{BP}}\mathrm{TiSp}(n^{1+\epsilon}, n^{\epsilon})$, in other words, proving that $\overrightarrow{\mathrm{BP}}\mathrm{TiSp}(n^{1+\epsilon}, n^{\epsilon}) \subseteq \Sigma_3\mathrm{Time}(o(n))$. Their result that $\Sigma_3\mathrm{Time}(n) \not\subseteq \overrightarrow{\mathrm{BP}}\mathrm{TiSp}(n^{1+\epsilon}, n^{\epsilon})$ then follows by a standard time hierarchy for alternating time.[13] To prove this simulation one argues as follows. First one *derandomizes* the $\overrightarrow{\mathrm{BP}}\mathrm{TiSp}(n^{1+\epsilon}, n^{\epsilon})$ machine using a pseudorandom generator by Nisan [Nis1] that has seed length n^{ϵ}. This gives $\overrightarrow{\mathrm{BP}}\mathrm{TiSp}(n^{1+\epsilon}, n^{\epsilon}) \subseteq \mathrm{BP}^{n^{\epsilon}}\mathrm{TiSp}(n^{1+\epsilon}, n^{\epsilon})$, where '$\mathrm{BP}^{n^{\epsilon}}$' means 'using at most n^{ϵ} random bits.' (Note we do not specify anymore if the machine has one-way or two-way access to its random bits, because now the machine simply could copy its n^{ϵ} random bits onto a random-access work tape.) Second, one replaces the n^{ϵ} random bits by two quantifiers, using Lautemann's result [Lau] discussed in Section 1. For this replacement, Lautemann's result needs to quantify over $(n^{\epsilon})^2$ bits, and thus one obtains $\mathrm{BP}^{n^{\epsilon}}\mathrm{TiSp}(n^{1+\epsilon}, n^{\epsilon}) \subseteq \exists^{n^{2\epsilon}}\forall^{n^{2\epsilon}}\mathrm{TiSp}\left(n^{1+O(\epsilon)}, n^{\epsilon}\right)$. By using another quantifier to speed up the running time of the simulation (cf. [vM]), one gets $\overrightarrow{\mathrm{BP}}\mathrm{TiSp}(n^{1+\epsilon}, n^{\epsilon}) \subseteq \Sigma_3\mathrm{Time}(o(n))$.

We now explain the difficulties we encounter in the proof of our result that $\Sigma_3\mathrm{Time}(n) \not\subseteq \overleftrightarrow{\mathrm{BP}}\mathrm{TiSp}\left(n^{1+o(1)}, n^{1-\epsilon}\right)$. Following the above outline, our task is to show the following simulation: $\overleftrightarrow{\mathrm{BP}}\mathrm{TiSp}(n^{1+\epsilon}, n^{1-\epsilon}) \subseteq \Sigma_3\mathrm{Time}(o(n))$. The main difficulty that we face in this extension to machines with two-way access to random bits is that in this setting only generators much weaker than Nisan's [Nis1] are known. Specifically, we use a generator by Impagliazzo et al. [INW], which raises two problems. The first problem is that, regardless of the amount of space used by the machine, this generator always has seed length $|\sigma| \gg \sqrt{n}$ (as opposed to n^{ϵ}). This is problematic because, as explained above, the approach in [DvM] is to replace the random bits for the seed σ of the generator by alternations. However, Lautemann's approach would need to quantify over $|\sigma|^2 \gg n$ bits, which would prevent us from obtaining a sublinear-time simulation. We solve this problem using our quasilinear-time simulation of probabilistic time (Theorem 3) instead of Lautemann's simulation.

The second problem is that we do not know how to compute the generator of [INW] in less than time $|\sigma|^2 \gg n$. Again, this seems to prevent us from obtaining a sublinear-time simulation. In fact, the argument in [DvM] directly exploits the fact that Nisan's generator [Nis1] is computable quickly. We solve this problem as follows. First we notice that the generator in [INW] can be computed time-efficiently *using alternations*. Then we apply our Corollary 4, which shows that our assumption ($\Sigma_3\mathrm{Time}(n) \subseteq \overleftrightarrow{\mathrm{BP}}\mathrm{TiSp}\left(n^{1+o(1)}, n^{1-\epsilon}\right) \subseteq \mathrm{BPTime}\left(n^{1+o(1)}\right)$) implies a time-efficient collapse of the alternating-time hierarchy to the third level.

We now restate our time-space lower bound and then prove it.

Definition (6, restated)**.** *We denote by* $\overleftrightarrow{\mathrm{BP}}\mathrm{TiSp}(t, s)$ *the set of languages accepted by probabilistic Turing machines, with two-sided error, that run simultaneously in time t and space s, with random access to input and work tapes, and two-way sequential access to the random-bit tape.*

Theorem (7, restated)**.** *For every constant $\epsilon > 0$,* $\Sigma_3\mathrm{Time}(n) \not\subseteq \overleftrightarrow{\mathrm{BP}}\mathrm{TiSp}\left(n^{1+o(1)}, n^{1-\epsilon}\right)$.

[13] This is a simplification of the techniques in [DvM]: in [DvM] they obtain the stronger result that $\Sigma_3\mathrm{Time}(n) \not\subseteq \overrightarrow{\mathrm{BP}}\mathrm{TiSp}\left(n^{3-\epsilon}, n^{\epsilon}\right)$ using a more involved argument.

We use the following generator by Impagliazzo, Nisan, and Wigderson.

Lemma 17 ([INW], Theorem 4)**.** *For every sufficiently small constant $\epsilon > 0$ and sufficiently large n, there exists a pseudorandom generator $G : \{0,1\}^{m^{1-\delta}} \to \{0,1\}^{m^{1+\delta}}$, with $m = m(n) := n^2$ and $\delta = \epsilon/4$, that satisfies the following:*

- Pseudorandomness: *Let M be a machine in $\overleftrightarrow{\mathrm{BP}}\mathrm{TiSp}\left(t, m^{1-\epsilon}\right)$ where $t = t(n) = m^{1+o(1)}$. For every fixed $x \in \{0,1\}^n$, $\left|\Pr_{u\in\{0,1\}^t}[M(x;u) = 1] - \Pr_{\sigma\in\{0,1\}^{m^{1-\delta}}}[M(x;G(\sigma)) = 1]\right| \le$* 1/9.
- Complexity: *Given a seed $\sigma \in \{0,1\}^{m^{1-\delta}}$ and the index $i \le m^{1+\delta}$ to an output bit, the i-th output bit of $G(\sigma)$ is computable simultaneously in time* $\mathrm{poly}(m)$ *and space* $O(m^{1-\delta})$.

Remark 18 (Remark on the proof of Lemma 17)**.** *In [INW] they sketch a proof that one can fool Turing machines running simultaneously in time t and space s using a generator G with seed length $O(\sqrt{t \cdot s} \cdot \log t)$. We now explain why this implies Lemma 17 above.*

First note that in our case $t = t(n) = m^{1+o(1)}$ and $s = m^{1-\epsilon}$. When $t \le m^{1+\delta}$ then the generator in [INW] has seed length $O(\sqrt{m^{1+\delta+1-\epsilon}} \cdot \log m) = O(m^{1-3\cdot\epsilon/8} \cdot \log m) \le m^{1-\epsilon/4} = m^{1-\delta}$ for sufficiently large m.

In [INW], they only argue that the generator fools Turing machines (as opposed to $\overleftrightarrow{\mathrm{BP}}\mathrm{TiSp}\,(t,s)$). However, their proof is only exploiting that the machine has sequential access to the random-bit tape, and that the state of the machine can be described by s bits, both of which hold in our model $\overleftrightarrow{\mathrm{BP}}\mathrm{TiSp}\,(t,s)$. In particular, their proof does not rely on Turing machines being a uniform model of computation, and thus their generator works even after we hardwire an arbitrary input x.

Finally, the authors of [INW] claim without proof (Section 4 in [INW]) that the generator can be computed "in polynomial time and polylog space in the size of the output of the generator." We do not see how to do that when the seed is polynomially related to the size of the output of the generator. However, it is not too hard to see that, given a seed, the generator is computable simultaneously in polynomial time and linear space (details omitted).

Notation for the proof of Theorem 7: The proof of Theorem 7 involves a series of inclusions between complexity classes. To reason about these classes, it is convenient to give the following definitions. We denote by $\mathrm{TiSp}\,(t,s)$ the set of (languages accepted by) Turing machines running simultaneously in time t and space s, with random access to input and work tapes (cf. [vM, FLvMV]). For a complexity class $\mathcal{C}$ (e.g. $\mathrm{TiSp}\,(t,s)$) and a function $f = f(n)$ we define the class $\mathrm{BP}^f\mathcal{C}$ to be the set of languages L for which there exists $M(x;u) \in \mathcal{C}$ such that $x \in L \Rightarrow \Pr_{u\in\{0,1\}^f}[M(x;u) = 1] \ge 2/3$, and $x \notin L \Rightarrow \Pr_{u\in\{0,1\}^f}[M(x;u) = 1] \le 1/3$, where the complexity of M is measured in terms of $|x|$ (as opposed to $|x| + |u|$). We analogously define $\exists^f\mathcal{C}$ (namely $x \in L \Leftrightarrow \exists u \in \{0,1\}^f : M(x;u) = 1$), and $\forall^f\mathcal{C}$.

Proof of Theorem 7. We assume that $\Sigma_3\mathrm{Time}\,(n) \subseteq \overleftrightarrow{\mathrm{BP}}\mathrm{TiSp}\left(n^{1+o(1)}, n^{1-\epsilon}\right)$, and derive a contradiction to the time hierarchy for alternating time, namely $\Sigma_3\mathrm{Time}\,(m) \subseteq$

$\Sigma_3\mathrm{Time}\left(o(m)\right)$ (see, e.g., Section 3.1 in [vM]). Let $m = m(n) := n^2$ (any $m = n^{1+\Omega(1)}$ would do). We have the following contradiction:

$$\begin{aligned}
\Sigma_3\mathrm{Time}\left(m\right) &\subseteq \overleftrightarrow{\mathrm{BP}}\mathrm{TiSp}\left(m^{1+o(1)}, m^{1-\epsilon}\right) && (5)\\
&\subseteq \mathrm{BP}^{m^{1-\delta}}\mathrm{TiSp}\left(\mathrm{poly}(m), m^{1-\delta}\right) && (6)\\
&\subseteq \exists^{m^{1-\delta/2}}\forall^{m^{1-\delta/2}}\exists^{m^{1-\delta/2}}\mathrm{TiSp}\left(\mathrm{poly}(m), m^{1-\delta/2}\right) && (7)\\
&\subseteq \Sigma_{O(1)}\mathrm{Time}\left(m^{1-\delta/4}\right) && (8)\\
&\subseteq \Sigma_3\mathrm{Time}\left(o(m)\right) && (9)\\
&\ \text{contradiction.} && (10)
\end{aligned}$$

Inclusion (5) holds by assumption plus a padding argument (see, e.g., [Pap]).

Inclusion (6) follows by using the INW generator from Lemma 17. More specifically, let M be a machine in $\overleftrightarrow{\mathrm{BP}}\mathrm{TiSp}\left(m^{1+o(1)}, m^{1-\epsilon}\right)$ and let $G : \{0,1\}^{m^{1-\delta}} \to \{0,1\}^{m^{1+\delta}}$ be the generator from Lemma 17. Now consider the machine M' that uses $m^{1-\delta}$ random bits σ, and simulates the machine M, but whenever M accesses the i-th random bit, M' computes on the fly the i-th output bit of $G(\sigma)$ and uses that instead. By reusing the same space to compute $G(\sigma)$ every time M accesses a random bit, and because G is computable simultaneously in time $\mathrm{poly}(m)$ and space $O(m^{1-\delta})$ by Lemma 17, we have that $M' \in \mathrm{BP}^{m^{1-\delta}}\mathrm{TiSp}\left(\mathrm{poly}(m), m^{1-\delta}\right)$. Also, for every input x we have $M(x) = M'(x)$ by the pseudorandomness property of INW in Lemma 17 (assuming without loss of generality that the error probability of M is a sufficiently small constant).

Inclusion (7) follows by (a variant of) Theorem 3. Specifically, it is easy to check that Theorem 3 can be extended to obtain the following inclusion, which implies Inclusion (7): For any $t = t(n)$ and $r = r(n) \le t(n)$ polynomially related to n, we have

$$\mathrm{BP}^{r}\mathrm{TiSp}\left(t, r\right) \subseteq \exists^{r'}\forall^{r'}\exists^{r'}\mathrm{TiSp}\left(t', r'\right),$$

where $r' = r \cdot \mathrm{poly}\log(n)$ and $t' = t \cdot \mathrm{poly}\log(n)$.

Inclusion (8) follows by the fact, usually credited to [Nep], that one can trade alternations for time in sublinear-space computations. More formally, one can prove the following inclusion (see e.g. [vM], Section 3.2):

$$\mathrm{TiSp}\left(t, s\right) \subseteq \Sigma_{2k}\mathrm{Time}\left(\left(t \cdot s^k\right)^{1/(k+1)}\right).$$

(The number of alternations in the above inclusion can be reduced to $k+1$ from $2k$ [FLvMV]. This gives a better constant for the $\Omega(1)$ term in our lower bound $n^{1+\Omega(1)}$.)

Inclusion (9) follows by a collapse similar to Corollary 4. Specifically, by assumption we have that $\Sigma_3\mathrm{Time}\left(n\right) \subseteq \overleftrightarrow{\mathrm{BP}}\mathrm{TiSp}\left(n^{1+o(1)}, n^{1-\epsilon}\right) \subseteq \mathrm{BPTime}\left(n^{1+o(1)}\right)$. Since probabilistic time is closed under complement, by Theorem 3 we get that $\mathrm{BPTime}\left(n^{1+o(1)}\right) \subseteq \Pi_3\mathrm{Time}\left(n^{1+o(1)}\right)$. Combining the two things we get

$$\Sigma_3\mathrm{Time}\left(n\right) \subseteq \Pi_3\mathrm{Time}\left(n^{1+o(1)}\right). \qquad (11)$$

Intuitively, this means that whenever we have a computation with three quantifiers we can complement them only paying a subpolynomial blow up in the running time, which gives the collapse. More formally, consider a computation in $\Sigma_{O(1)}\mathrm{Time}\left(m^{1-\delta/4}\right)$. We can assume that $1-\delta/4 \geq 2/3$ without loss of generality, and thus the $\Sigma_{O(1)}\mathrm{Time}\left(m^{1-\delta/4}\right)$ machine runs in time $m^{1-\delta/4} \geq n^{4/3} \geq n$. Consequently, Equation (11) implies that

$$\Sigma_3\mathrm{Time}\left(m^{1-\delta/4}\right) \subseteq \Pi_3\mathrm{Time}\left((m^{1-\delta/4})^{1+o(1)}\right)$$

by padding. Applying this padded Equation (11) a constant number of times and collapsing adjacent quantifiers, we obtain that $\Sigma_{O(1)}\mathrm{Time}\left(m^{1-\delta/4}\right)$ collapses to $\Sigma_3\mathrm{Time}\,(m')$, where m' is $m^{1-\delta/4}$ raised, a constant number of times, to an exponent that is $1 + o(1)$, and so $m' = m^{(1-\delta/4)(1+o(1))} = m^{1-\Omega(1)} = o(m)$.

The contradiction (10) is obtained by diagonalization (see, e.g., Section 3.1 in [vM]). □

Remark 19 (On the difficulty of proving similar lower bounds for functions below Σ_3.)**.** *We note that a key step in the proof of our lower bounds (theorems 7 and 8) is the inclusion* $\Sigma_{O(1)}\mathrm{Time}\,(t) \subseteq \Sigma_3\mathrm{Time}\left(t^{1+o(1)}\right)$ *under the assumption that* $\Sigma_3\mathrm{Time}\,(n) \subseteq \mathrm{BPTime}\left(n^{1+o(1)}\right)$*. As we have seen, this result relies on our time-efficient simulation of probabilistic time with three quantifiers (Theorem 3); cf. the proof of Theorem 7. Since we do not expect to prove a time-efficient simulation of probabilistic time using only two quantifiers (cf. Theorem 5), we also do not expect to prove* $\Sigma_{O(1)}\mathrm{Time}\,(t) \subseteq \Sigma_2\mathrm{Time}\left(t^{1+o(1)}\right)$ *under the assumption that* $\Sigma_2\mathrm{Time}\,(n) \subseteq \mathrm{BPTime}\left(n^{1+o(1)}\right)$*. This is the obstacle to obtaining a lower bound for a function in* $\Sigma_2\mathrm{Time}\,(n \cdot \mathrm{poly}\log n)$*, and in particular for* SAT*.*

6 Open Problems

In this section we list a few open problems.

(1) Prove an $n^{1+\Omega(1)}$ time lower bound on probabilistic Turing machines using space $O(\log n)$ with random access to both the input tape and the random-bit tape (for a function computable in linear space). Our results (Theorem 7) only apply to models with sequential access to the random-bit tape.

(2) Prove a superlinear time lower bound on probabilistic Turing machines with random-access to the input, two-way sequential access to the work tape with no space restrictions, and one-way access to the random-bit tape (for a function computable in linear space). This would be the "natural" probabilistic extension of the classic result in [MS]; our results (Theorem 8) only apply to the weaker model where the work tape is initialized with random bits.

Acknowledgements. Scott Diehl and Dieter van Melkebeek obtained Theorem 3 independently from us (personal communication, Oct. 2005). We are grateful to Dieter van Melkebeek for an email exchange about time-space lower bounds, Turing machine models, and the results in [DvM, vMR], and for comments on the write-up. Many thanks to Salil Vadhan for helpful discussions and comments on the write-up. We also would like to thank Alex Healy for helpful discussions, in particular on Nisan's generator [Nis1].

References

[Ajt1] M. Ajtai. Σ_1^1-formulae on finite structures. *Ann. Pure Appl. Logic*, 24(1):1–48, 1983.

[Ajt2] M. Ajtai. Approximate counting with uniform constant-depth circuits. In *Advances in computational complexity theory (New Brunswick, NJ, 1990)*, pages 1–20. Amer. Math. Soc., Providence, RI, 1993.

[ABO] M. Ajtai and M. Ben-Or. A Theorem on Probabilistic Constant Depth Computation. In ACM, editor, *Proceedings of the sixteenth annual ACM Symposium on Theory of Computing, Washington, DC, April 30–May 2, 1984*, pages 471–474, 1984.

[AKR$^+$] E. Allender, M. Koucký, D. Ronneburger, S. Roy, and V. Vinay. Time-Space Tradeoffs in the Counting Hierarchy. In *Proceedings of the Sixteenth Annual Conference on Computational Complexity*, pages 295–302. IEEE, June 18–21 2001.

[AGHP] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures & Algorithms*, 3(3):289–304, 1992.

[BDG] J. L. Balcazar, J. Diaz, and J. Gabarro. *Structural complexity Theory.* Springer-Verlag New York, Inc., New York, NY, USA, 1988.

[BSSV] P. Beame, M. Saks, X. Sun, and E. Vee. Time-space trade-off lower bounds for randomized computation of decision problems. *J. ACM*, 50(2):154–195 (electronic), 2003.

[Can] R. Canetti. More on BPP and the polynomial-time hierarchy. *Inform. Process. Lett.*, 57(5):237–241, 1996.

[CR] S. Chaudhuri and J. Radhakrishnan. Deterministic Restrictions in Circuit Complexity. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 30–36, Philadelphia, Pennsylvania, 22–24 May 1996.

[CLRS] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms.* MIT Press, Cambridge, MA, second edition, 2001.

[DvM] S. Diehl and D. van Melkebeek. Time-space lower bounds for the polynomial-time hierarchy on randomized machines. *SIAM J. Comput.*, 36(3):563–594, 2006.

[ESY] S. Even, A. L. Selman, and Y. Yacobi. The complexity of promise problems with applications to public-key cryptography. *Inform. and Control*, 61(2):159–173, 1984.

[FLvMV] L. Fortnow, R. Lipton, D. van Melkebeek, and A. Viglas. Time-Space Lower Bounds for Satisfiability. To appear in Journal of the ACM, Available from *http://www.cs.wisc.edu/∽dieter/Research/sat-ts.html*, 2005.

[FSS] M. L. Furst, J. B. Saxe, and M. Sipser. Parity, Circuits, and the Polynomial-Time Hierarchy. *Mathematical Systems Theory*, 17(1):13–27, April 1984.

[GG] O. Gabber and Z. Galil. Explicit constructions of linear size superconcentrators. *Journal of Computer and System Sciences*, 22:407–420, 1981.

[Gil] D. Gillman. A Chernoff bound for random walks on expander graphs. *SIAM J. Comput.*, 27(4):1203–1220 (electronic), 1998.

[Gol] O. Goldreich. *Modern cryptography, probabilistic proofs and pseudorandomness*, volume 17 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 1999.

[GZ] O. Goldreich and D. Zuckerman. Another proof that BPP $\subseteq$ PH (and more). *Electronic Colloquium on Computational Complexity*, Technical Report TR97-045, September 1997. `http://www.eccc.uni-trier.de/eccc`.

[Hås] J. Håstad. *Computational limitations of small-depth circuits*. MIT Press, 1987.

[INW] R. Impagliazzo, N. Nisan, and A. Wigderson. Pseudorandomness for Network Algorithms. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 356–364, Montréal, Québec, Canada, 23–25 May 1994.

[JM] S. Jimbo and A. Maruoka. Expanders obtained from affine transformations. *Combinatorica*, 7(4):343–355, 1987.

[Kah] N. Kahale. Eigenvalues and expansion of regular graphs. *J. Assoc. Comput. Mach.*, 42(5):1091–1106, 1995.

[KN] E. Kushilevitz and N. Nisan. *Communication complexity*. Cambridge University Press, Cambridge, 1997.

[Lau] C. Lautemann. BPP and the polynomial hierarchy. *Inform. Process. Lett.*, 17(4):215–217, 1983.

[MS] W. Maass and A. Schorr. Speed-up of Turing machines with one work tape and a two-way input tape. *SIAM J. Comput.*, 16(1):195–202, 1987.

[Mar] G. A. Margulis. Explicit construction of concentrator. *Problems Inform. Transmission*, 9:325–332, 1973.

[NN] J. Naor and M. Naor. Small-bias probability spaces: efficient constructions and applications. In *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing*, pages 213–223, 1990.

[Nep] V. A. Nepomnjaščiĭ. Rudimentary predicates and Turing calculations. *Soviet Mathematics-Doklady*, 11(6):1462–1465, 1970.

[Nis1] N. Nisan. Pseudorandom Generators for Space-bounded Computation. *Combinatorica*, 12(4):449–461, 1992.

[Nis2] N. Nisan. On read-once vs. multiple access to randomness in logspace. *Theoret. Comput. Sci.*, 107(1):135–144, 1993. Structure in complexity theory (Barcelona, 1990).

[Pap] C. H. Papadimitriou. *Computational Complexity*. Addison–Wesley, 1994.

[Raz] A. Razborov. Pseudorandom Generators Hard for k-DNF Resolution and Polynomial Calculus Resolution, 2002-2003. Manuscript. Available from `http://http://www.mi.ras.ru/∽razborov/`.

[RS] A. Russell and R. Sundaram. Symmetric alternation captures BPP. *Comput. Complexity*, 7(2):152–162, 1998.

[SV] S. Sanghvi and S. Vadhan. The round complexity of two-party random selection. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 338–347, New York, NY, USA, 2005. ACM Press.

[SBI] N. Segerlind, S. Buss, and R. Impagliazzo. A switching lemma for small restrictions and lower bounds for k-DNF resolution. *SIAM J. Comput.*, 33(5):1171–1200 (electronic), 2004.

[Sip] M. Sipser. A Complexity Theoretic Approach to Randomness. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, pages 330–335, Boston, Massachusetts, 25–27 Apr. 1983.

[Sto] L. Stockmeyer. On Approximation Algorithms for #P. *SIAM J. on Computing*, 14(4):849–861, Nov. 1985.

[vM] D. van Melkebeek. Time-Space Lower Bounds for NP-Complete Problems. In *Current Trends in Theoretical Computer Science*, pages 265–291. World Scientific, 2004.

[vMR] D. van Melkebeek and R. Raz. A Time Lower Bound for Satisfiability. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP)*, pages 971–982, 2004.

[Vio] E. Viola. *The Complexity of Hardness Amplification and Derandomization*. PhD thesis, Harvard University, 2006. `http://www.eccc.uni-trier.de/eccc`.

A Lower Bound on Stronger Computational Models

In this section we prove our time lower bound on two-tape Turing machines (Theorem 8). Rather than working directly in the model $\mathrm{BPTime}_1(t)$ (introduced in Section 1.4), which is incomparable to the previously considered space-bounded model (Def. 6), we proceed by extending the space-bounded model so that it includes $\mathrm{BPTime}_1(t)$ as a special case, and we prove a lower bound on this stronger model. Specifically, we extend $\overleftrightarrow{\mathrm{BP}}\mathrm{TiSp}(t,s)$ (Def. 6) by allowing machines to write on the random-bit tape, and denote this extension by $\overleftrightarrow{\mathrm{BP}}_w\mathrm{TiSp}(t,s)$. By ignoring the space-bounded random-access tapes, we see that $\overleftrightarrow{\mathrm{BP}}_w\mathrm{TiSp}(t,s)$ includes the model $\mathrm{BPTime}_1(t)$ as a special case. Let us formally define $\overleftrightarrow{\mathrm{BP}}_w\mathrm{TiSp}(t,s)$ and then state our lower bound.

Definition 20. *We denote by $\overleftrightarrow{\mathrm{BP}}_w\mathrm{TiSp}(t,s)$ the extension of $\overleftrightarrow{\mathrm{BP}}\mathrm{TiSp}(t,s)$ obtained by allowing machines to write on the random-bit tape. In other words, we denote by $\overleftrightarrow{\mathrm{BP}}_w\mathrm{TiSp}(t,s)$ the set of languages accepted by probabilistic Turing machines, with two-sided error, running in time t, which have the following tapes:*

- *a random-access input tape,*
- *several random-access work tapes for at most s bits, and*
- *one sequential-access two-way read-write tape that is initially filled with random bits.*

Theorem 21. *For every constant $\epsilon > 0$, $\Sigma_3\mathrm{Time}(n) \not\subseteq \overleftrightarrow{\mathrm{BP}}_w\mathrm{TiSp}\left(n^{1+o(1)}, n^{1-\epsilon}\right)$, and in particular $\mathrm{QSAT}_3 \notin \overleftrightarrow{\mathrm{BP}}_w\mathrm{TiSp}\left(n^{1+o(1)}, n^{1-\epsilon}\right)$.*

The outline of the proof of Theorem 21 is similar to that of the proof of Theorem 7 (cf. Section 5). The two main differences are: (1) we need to observe that the INW generator in Lemma 17 also works if the machine is allowed to write on the random-bit tape, and (2) we need to show that it is still possible to simulate the machine using small space (and quantifiers). The observation (1) is in Remark 22 below, and the simulation (2) is proved in Claim 23 using ideas from a simulation by Maass and Schorr [MS] which was recently rediscovered by van Melkebeek and Raz [vMR].

Remark 22 (On the pseudorandomness property of the INW generator)**.** *We point out that the INW generator in Lemma 17 also fools machines that are allowed to write on the sequential two-way random bits tape, i.e. $\overleftrightarrow{\mathrm{BP}}_w\mathrm{TiSp}(t,s)$ machines. While this extension is not explicitly stated in [INW], it can be obtained using the techniques in [INW]. Roughly, their communication-complexity simulation in the proof of Theorem 5 in [INW] (which is only stated for machines with one tape) can be extended to machines with several space-bounded work tapes, by having Alice and Bob also communicate all the contents of the random-access work tapes (details omitted).*

Proof of Theorem 21. The outline of the proof follows closely that of the proof of Theorem 7. Let $m = m(n) := n^2$. We have the following contradiction:

$$\begin{aligned}
\Sigma_3\mathrm{Time}\,(m) &\subseteq \overleftrightarrow{\mathrm{BP}}_w\mathrm{TiSp}\left(m^{1+o(1)}, m^{1-\epsilon}\right) && (12)\\
&\subseteq \mathrm{BP}^{m^{1-\Omega(1)}}\exists^{m^{1-\Omega(1)}}\mathrm{TiSp}\left(\mathrm{poly}(m), m^{1-\Omega(1)}\right) && (13)\\
&\subseteq \exists^{m^{1-\Omega(1)}}\forall^{m^{1-\Omega(1)}}\exists^{m^{1-\Omega(1)}}\mathrm{TiSp}\left(\mathrm{poly}(m), m^{1-\Omega(1)}\right) && (14)\\
&\subseteq \Sigma_{O(1)}\mathrm{Time}\left(m^{1-\Omega(1)}\right) && (15)\\
&\subseteq \Sigma_3\mathrm{Time}\,(o(m)) && (16)\\
&\quad\ \text{contradiction.} && (17)
\end{aligned}$$

Inclusions (12), (14)-(16), and the final contradiction 17 can be obtained exactly as in the proof of Theorem 7. The next claim proves Inclusion (13) and thus concludes the proof of the theorem.

Claim 23. *Let $M(x;u)$ be a machine in $\overleftrightarrow{\mathrm{BP}}_w\mathrm{TiSp}\left(m^{1+o(1)}, m^{1-\epsilon}\right)$ where $\epsilon > 0$ is any fixed constant and $m = m(n) := n^2$. Let $G : \{0,1\}^{m^{1-\delta}} \to \{0,1\}^{m^{1+\delta}}$ be the generator from Lemma 17. Then the language $\{(x;\sigma) : M(x;G(\sigma)) = 1\}$ is in $\exists^{m^{1-\Omega(1)}}\mathrm{TiSp}\left(\mathrm{poly}(m), m^{1-\Omega(1)}\right)$. In particular,*

$$\overleftrightarrow{\mathrm{BP}}_w\mathrm{TiSp}\left(m^{1+o(1)}, m^{1-\Omega(1)}\right) \subseteq \mathrm{BP}^{m^{1-\Omega(1)}}\exists^{m^{1-\Omega(1)}}\mathrm{TiSp}\left(\mathrm{poly}(m), m^{1-\Omega(1)}\right).$$

Proof. Let $t = m^{1+o(1)}$ be the running time of the machine M, $s = m^{1-\epsilon}$ the random-access space used by M, and let us denote by τ the (two-way) read-write sequential tape of M, which we will initially fill with the pseudorandom bits $G(\sigma)$. We assume that τ has tape squares numbered as $0, 1, 2, \ldots$, and that M starts with the head on tape square 0 (the same proof carries through for the case where τ has tape squares numbered as $\ldots, -2, -1, 0, 1, 2, \ldots$).

We use the idea of *crossing sequences*, (see, e.g., [vMR] for background). Let us divide τ in consecutive blocks where the first block is of size d and all the others are of size $b := m^{1-\epsilon/3}$. The parameter d will be specified later. Note b and d completely specify this subdivision in blocks. For fixed b, d, let us define a *crossing* to be a pair $(a \to a', S)$ where a is an index to a block in τ, $a' = a \pm 1$ and S completely specifies the state of the machine M *except* the contents of τ. More specifically, S specifies the content of the random-access tapes, the position of the head on the input tape, and the internal state of the machine. As usual, we think of crossing $(a \to a', S)$ as meaning that M is crossing the boundary of block a towards block a' while being in state S (but let us stress again that S does not specify the content of τ).

Consider the computation $M(x;G(\sigma))$ that runs in time t. Since different values of $d \in \{0, 1, \ldots, b-1\}$ give rise to disjoint sets of blocks, it is not hard to see that there must exist $d < b$ such that the number of crossings induced by the computation $M(x;G(\sigma))$ is at most $t/b = m^{1+o(1)-1+\epsilon/3} \leq m^{\epsilon/2}$.

Simulation: We simulate the machine M as follows: First we guess a shift d and a sequence of t/b crossings. Then we check consistency of the computation. For this, let us divide the space of the simulation in two parts: a current-block part of b bits and a current-state part of s bits (jumping ahead, note $b+s=m^{1-\epsilon/3}+m^{1-\epsilon}=m^{1-\Omega(1)}$).

As a base case, we initialize the bits in current-block with $G(\sigma)_0 \cdot G(\sigma)_1 \cdots G(\sigma)_{d-1}$, and we simulate the machine, using current-state as its random-access space and current-block as the first block of τ, until it crosses the block boundary towards block 1 (which starts at tape square d). (If the machine never crosses this boundary then the simulation is easy.) When that happens, we look at the first crossing $(a' \to a'', S')$ in the guessed list and check that $a'=0, a''=1$, and current-state equals S'.

Next, for every block number i, we proceed as follows. First we initialize the bits in current-block with $G(\sigma)_{d+ib} \cdot G(\sigma)_{d+ib+1} \cdots G(\sigma)_{d+ib+(b-1)}$. Then we scan, in order, the list of guessed crossings. Whenever we encounter a crossing of the form $(a \to i, S)$ we do the following. We copy S on current-state and we simulate M (using current-state as the random-access space of M) until it crosses the block boundary towards block a'. Then we look at the next crossing $(\alpha \to \alpha', S')$ in the guessed list and check that $\alpha = i$, $\alpha' = a'$ and current-state equals S'. We continue in this way until the list is over.

Finally, we check that the last crossing corresponds to the accepting state of M. (Without loss of generality we can assume that M, if it accepts, it does so by entering its accepting state while crossing a block boundary.)

Correctness: It is easy to verify that the simulation accepts if and only if M does.

Complexity: Note that each crossing can be specified by $O(m^{1-\epsilon})$ bits, and therefore we guess at most $O(m^{1-\epsilon+\epsilon/2}) = m^{1-\Omega(1)}$ bits total. The rest of the computation can be done simultaneously in time poly(m) and space $m^{1-\Omega(1)}$. To see this, note that the INW generator is computable in these resources by Lemma 17, while the rest of the simulation only uses space for the current-block part of b bits and the current-state part of s bits, which sum up to $b+s=m^{1-\epsilon/3}+m^{1-\epsilon}=m^{1-\Omega(1)}$. (The simulation is easily seen to run in polynomial time.)

The 'in particular' part of the claim follows by the first part of the claim plus the pseudorandomness property of the INW generator in Lemma 17 (cf. remark 22). (I.e. we use $m^{1-\Omega(1)}$ random bits for the seed σ of the INW generator G.) □

□

B On the Error Parameter

In this section we discuss the dependence of our results on the error probability ϵ of the BPTime(t) machines.

While we proved our results only for BPTime(t) machines with error $\epsilon = 1/3$, our results immediately generalize to any constant error $0 < \epsilon < 1/2$. In fact, they also apply to more general error parameters $\epsilon = \epsilon(t)$ as we now discuss. For this discussion, let $\epsilon-$ApprMaj denote the promise problem of computing Majority on an input bit string whose fraction of 1's is promised to be either at least $1-\epsilon$ or at most ϵ. (Note that the previously considered approximate majority equals $1/3-$ApprMaj.) For the case where the error ϵ is shrinking (e.g. $\epsilon := 1/t^2$) it is possible to generalize our Theorem 1 to show that computing $\epsilon-$ApprMaj on n bits requires bottom fan-in at least $\Omega(\log(n)/\log(1/\epsilon))$, which implies that any relativizing Σ_2 simulation of BPTime(t) must have running time at least $\Omega(t^2/\log(1/\epsilon))$ (e.g. $\Omega(t^2/\log t)$ when $\epsilon = 1/t^2$). In particular, polynomially small error does not give polynomial savings in the running time, and even in this case our results show that the running time of previous simulations [Sip, Lau] is optimal up to polylogarithmic factors for relativizing techniques.

On the other hand, for large error $\epsilon = 1/2 - \alpha$ where $\alpha = o(1)$, one can use Håstad's switching lemma [Hås] to argue that small depth-3 circuits for $\epsilon-$ApprMaj need bottom fan-in at least $\Omega(1/\alpha)$ (details omitted). Again, this implies that any relativizing Σ_2 simulation of BPTime(t) must have running time at least $\Omega(t/\alpha)$ (which is only better than Theorem 5 when $\alpha = o(1/t)$). This latter bound is met, up to a polynomial factor, by first applying standard error reduction techniques to bring the error down to, say, $\epsilon = 1/3$, and then plugging in previous results [Sip, Lau].

C Oracle Separation

In this section we present the proof that our circuit lower bound for approximate majority in Theorem 1 implies our oracle separation in Theorem 5. The intuition behind the proof was discussed at the end of Section 2.

Proof of Theorem 5 assuming Theorem 1. The overall structure of the proof follows standard arguments (see e.g., [FSS], or Theorem 14.5 in [Pap]). Fix a time bound $t = t(n)$. Consider the collection of oracles $\mathcal{A}$ where for every $\mathrm{A} \in \mathcal{A}$ and for every integer m, the fraction of strings of length m in the oracle A is at least $2/3$ or at most $1/3$. For $\mathrm{A} \in \mathcal{A}$ we consider the language

$$L^{\mathrm{A}} := \left\{ 0^n : \Pr_{x:|x|=t(n)}[x \in \mathrm{A}] \geq 2/3 \right\}.$$

For every $\mathrm{A} \in \mathcal{A}$ we have that $L^{\mathrm{A}} \in$ BPTime(t) with oracle access to A: on input 0^n, we simply query the oracle at a random x of length $t(n)$ and return its answer.

On the other hand, we show that there is $\mathrm{A} \in \mathcal{A}$ such that $L^{\mathrm{A}} \notin \Sigma_2\mathrm{Time}(o(t^2))$ with oracle access to A. The construction of A proceeds by diagonalization. Consider an enumeration $M_1, M_2, \ldots$ of $\Sigma_2\mathrm{Time}(o(t^2))$ machines. We assume that these machines only make oracle queries of length $t(n)$, on input of length n. This assumption can be removed by standard arguments, e.g. by considering 'sufficiently sparse' input lengths. We make this assumption because it simplifies the proof while preserving all the main ideas, and also because it holds for all known simulations (in particular all those in Table 2).

At stage i we fix a finite initial segment of the oracle A in such a way that $L^{\mathrm{A}} \neq M_i^{\mathrm{A}}$. Stage i works as follows. Let M_i run in time $t' = o(t^2)$. By standard techniques from [FSS] the machine M_i gives rise to an unbounded fan-in circuit of depth 3 as follows. The input to the circuit is the truth table of length $T = T(n) := 2^t$ of the oracle (as well as the bit-wise complement of this truth table). The top and the middle fan-ins of the circuit are $2^{O(t'(n))}$

(where the top fan-in is defined as the fan-in of the output gate), while the bottom fan-in is $k = k(n) := t'(n)/t(n) = o(t^2/t) = o(t)$. The bottom fan-in corresponds to the maximum number of oracle queries the machine makes on any computation path. Since each query is of length t and the oracle tape is erased after each query (see, e.g., [BDG]), the machine can only ask $t'(n)/t(n)$ queries.

We are now in the apply our Theorem 1. For sufficiently large n, a circuit with the above parameters cannot compute approximate majority on $T = 2^t$ bits. (Note that the bottom fan-in of the circuit is $k = o(t) = o(\log T)$.) Therefore we can augment the oracle A in such a way that $L^{\mathrm{A}} \neq M_i^{\mathrm{A}}$, while still keeping $\mathrm{A} \in \mathcal{A}$. $\square$