Graph Nonisomorphism Has Subexponential Size Proofs Unless The Polynomial-Time Hierarchy Collapses*

Adam R. Klivans†
Department of Mathematics, MIT
Cambridge, MA 02139
klivans@math.mit.edu

Dieter van Melkebeek‡
Department of Computer Sciences
University of Wisconsin
Madison, WI 53706
dieter@cs.wisc.edu

February 26, 2002

Abstract

Traditional hardness versus randomness results focus on time-efficient randomized decision procedures. We generalize these trade-offs to a much wider class of randomized processes. We work out various applications, most notably to derandomizing Arthur-Merlin games. We show that every language with a bounded round Arthur-Merlin game has subexponential size membership proofs for infinitely many input lengths unless exponential time coincides with the third level of the polynomial-time hierarchy (and hence the polynomial-time hierarchy collapses). Since the graph nonisomorphism problem has a bounded round Arthur-Merlin game, this provides the first strong evidence that graph nonisomorphism has subexponential size proofs.

We also establish hardness versus randomness tradeoffs for space bounded computation.

---

*A preliminary version of this paper appeared in the *Proceedings of the 31st ACM Symposium on the Theory of Computing* [KvM99].

†Supported in part by NSF grant CCR-9701304.

‡Supported in part by NSF grants CCR-9732922 and CCR-0133693.

1 Introduction

Randomness has proven to be a valuable tool in computer science. It is an indispensable resource in areas like cryptography and distributed computing. The situation is less clear for algorithms. Several computational problems turn out to have simple randomized solutions that require less time or space than the known deterministic algorithms. It is conceivable, however, that efficient deterministic algorithms exist for all of these problems.

In fact, many complexity theorists believe that randomness *cannot* significantly reduce the time or space complexity of a problem. To this end they try to construct efficient pseudo-random generators: deterministic procedures that stretch seeds containing a few random bits into much longer bit sequences that look random to the randomized process in question. This allows us to reduce the amount of randomness needed and, in some cases, eliminate the need for random bits completely without increasing the need for other resources by much.

Considerable progress has been made in the construction of pseudo-random generators that fool arbitrary time-bounded decision procedures. We now know how to construct them out of certain types of "hard" computational problems. The existence of such hard problems is still open. These constructions are often referred to as *hardness versus randomness trade-offs*; they rule out the possibility that both hard problems exist and randomness speeds up some time-bounded decision problem.

1.1 Traditional Hardness versus Randomness Trade-Offs

Blum and Micali [BM84] and Yao [Yao82] were the first to realize this connection. Nisan and Wigderson [NW94] established a whole range of hardness versus randomness trade-offs using an average-case hardness notion. They showed how to use any language in exponential time that requires large circuits to be computed exactly on slightly more than half of the inputs, to construct a pseudo-random generator that fools circuits of polynomial size. They obtained nontrivial derandomizations of polynomial-time randomized decision algorithms under average-case hardness assumptions and even deterministic polynomial-time simulations under the strongest of their hypotheses. Babai, Fortnow, Nisan, and Wigderson [BFNW93] and Impagliazzo and Wigderson [IW97] relaxed the hardness condition from an average-case one to a worst-case one. For their constructions they only had to start from a language in exponential time that requires large circuits to be computed exactly on *every* input. As a corollary, they showed how to simulate every polynomial-time randomized decision algorithm deterministically in subexponential time for infinitely many input lengths unless exponential time collapses to the second level of the polynomial-time hierarchy (and in fact to the class MA) [BFNW93].

The authors of these papers [NW94, BFNW93, IW97] used their techniques to derandomize traditional models of randomized computation, most notably BPP. We show how to apply the techniques used to derandomize BPP to more general models of randomized computation. The key observation we make is that the reductions used in the proofs are "black-box", i.e., they relativize. More specifically, the above papers start from a Boolean function f and construct a proposed pseudo-random generator based on it. They then argue that any small circuit that distinguishes the output of the proposed pseudo-random generator from the uniform distribution can be transformed into a small circuit that computes f. We observe that these reductions work for *any* nonuniform model of computation that satisfies certain closure properties. In particular, they work for oracle circuits given any fixed oracle B, i.e., for circuits consisting of AND, OR, and NOT gates, and gates that evaluate the oracle B. Thus, in order to build a pseudo-random generator that looks random to any small B-oracle circuit, we need only assume the existence of a function f that cannot be computed by small B-oracle circuits. The same holds for more recent constructions by Sudan et al. [STV00] that provide alternate proofs for some of the earlier results [NW94, BFNW93, IW97].

1.2 Derandomizing Arthur-Merlin Games

The above observations allow us to apply the classical hardness versus randomness results to various settings, in particular to the nondeterministic setting of Arthur-Merlin games [Bab85].

Arthur-Merlin games are verification procedures in which a prover (Merlin) tries to convince a randomized polynomial-time verifier (Arthur) of some fact, say that a given input belongs to some language. Arthur-Merlin games define an extension of the classical proof paradigm in two ways: the use of randomness by the verifier and the interaction between the prover and the verifier. Of particular interest is the class AM of languages for which there exists an Arthur-Merlin game with a bounded number of rounds of interaction between the verifier and the prover. The class AM forms a randomized extension of NP. The most notable problem in AM not known to be in NP is Graph Nonisomorphism [GMW91, GS89, BM88].

Derandomizing Arthur-Merlin games requires security against the nondeterministic adversary Merlin. Rudich [Rud97] pointed out that pseudo-random generators in the traditional cryptographic setting, where an adversary has more resources than the generator, cannot hope to have this property. This is because a nondeterministic adversary can guess the seed the verifier will use and run the generator on this seed to obtain the pseudo-random sequence the verifier will use. From a derandomization perspective, however, we can limit the resources the adversary can use. Nisan and Wigderson exploited this fact in the BPP setting. We use it in the context of Arthur-Merlin games. The drawback of cryptographic pseudo-random generators does not apply because the adversaries do not have the resources to run the generator – even if they correctly guess the seed.

We give evidence that AM coincides with NP. More specifically, we show that the existence of an exponential-time decidable language with high worst-case SAT-oracle complexity implies nontrivial derandomizations of AM. Examples of the trade-offs we obtain are presented in Table 1, where we use C^B to denote circuit complexity given access to oracle B. We provide the precise definitions in Section 2, and the general trade-off statement in Section 3.

hardness assumption:	derandomization consequence:
$\exists\, f \in \mathsf{NEXP} \cap \mathsf{coNEXP} : C_f^{\mathrm{SAT}}(n) = n^{\omega(1)}$	$\mathsf{AM} \subseteq \cap_{\epsilon>0}\mathsf{NTIME}[2^{n^\epsilon}]$
$\exists\, f \in \mathsf{NEXP} \cap \mathsf{coNEXP} : C_f^{\mathrm{SAT}}(n) = 2^{n^{\Omega(1)}}$	$\mathsf{AM} \subseteq \cup_{c>0}\mathsf{NTIME}[2^{(\log n)^c}]$
$\exists\, f \in \mathsf{NE} \cap \mathsf{coNE} : C_f^{\mathrm{SAT}}(n) = 2^{\Omega(n)}$	$\mathsf{AM} = \mathsf{NP}$

Table 1: Hardness versus randomness trade-offs for AM.

For example, the first line of Table 1 states that if there exists a function computable in $\mathsf{NEXP} \cap \mathsf{coNEXP}$ which requires SAT-oracle circuits of superpolynomial size for almost all input lengths then then AM is contained in nondeterministic subexponential time, i.e., has subexponential size proofs. The third line of Table 1 states that if we go further and assume there exists a function in $\mathsf{NE} \cap \mathsf{coNE}$ which requires SAT-oracle circuits of size $2^{\epsilon n}$ for some $\epsilon > 0$ and almost all input lengths then every language in AM has polynomial size proofs.

If the hardness condition on the left-hand side of Table 1 holds for infinitely many input lengths, then the corresponding derandomization on the right-hand side works for infinitely many input lengths. We refer to the *weak* version of Table 1 when we assume the above hardness conditions only hold for infinitely many input lengths. As in the above papers [NW94, BFNW93, IW97], we typically state our theorems assuming the hardness conditions are true for every input length. Both interpretations hold for all of our results.

We can view the assumptions in Table 1 as statements concerning the relationships among computation, nonuniformity, and nondeterminism. For example, the third entry in the table states that if nonuniformity and nondeterminism cannot significantly speed up computation then we can derandomize AM. We point out that if the hardness assumption in the first row of the weak version of Table 1 fails, then exponential time collapses to the third level of the polynomial-time hierarchy.

Arvind and Köbler [AK97] obtained similar results to those in Table 1 using *nondeterministic* circuits, but needed average-case hardness assumptions instead of worst-case ones. Nondeterministic circuits consist of AND, OR, and NOT gates, and have some additional input bits y. An input x is accepted if there is a setting of the additional input bits y that makes the circuit accept x. Nondeterministic circuits can be efficiently simulated by SAT-oracle circuits but presumably not the other way around. Nondeterministic circuits do not seem to have the closure properties that allow us to relax the hardness hypothesis from average-case to worst-case, whereas SAT-oracle circuits do. Bridging the gap between average-case and worst-case assumptions is crucial. It allows us to conclude that every language in AM, and graph nonisomorphism in particular, has subexponential size proofs for infinitely many input lengths unless the polynomial-time hierarchy collapses. See the weak version of the first line in Table 1. Without any unproven hypothesis, the smallest proofs known for the nonisomorphism of two graphs on n vertices are of size $2^{O(\sqrt{n \log n})}$, namely the transcripts of the deterministic graph isomorphism algorithm by Babai et al. [BL83, BKL83].

Recently, Miltersen and Vinodchandran [MV99] managed to relax the hardness assumption in the last line of Table 1. They showed that AM collapses to NP if there exists a Boolean function in $\mathsf{NE} \cap \mathsf{coNE}$ with exponential nondeterministic circuit complexity (instead of SAT-oracle circuit complexity). Their technique only works under strong hardness assumptions. In particular, it does not apply to the rest of Table 1. See Section 6 for a more detailed discussion.

1.3 Generalizing Hardness versus Randomness Trade-Offs

Our simulations of AM are a special case of a general derandomization tool which applies to any randomized process for which we can efficiently check whether a particular choice of random bits results in a successful computation. For example, for a fixed BPP algorithm and input, at least 2/3 of the random strings result in a "successful" computation. Any randomized function computation with this property, regardless of whether it involves a decision problem or not, falls within our framework. We formally define the notion of a success predicate in Section 4. If we can decide the success predicate of a randomized process with polynomial size B-oracle circuits then assuming the existence of a function with high B-oracle circuit complexity implies the existence of a pseudo-random generator suitable for derandomizing the process. Table 2 gives a summary of these trade-offs. The symbol $\mathcal{A}$ in Table 2 represents an arbitrary class of oracles and determines the complexity of computing the generator G.

hardness assumption:	complexity of G:	seed length:
$\exists\, f \in \mathsf{EXP}^{\mathcal{A}} : C_f^B(n) = n^{\omega(1)}$	$\mathsf{EXP}^{\mathcal{A}}$	$O(n^{\epsilon})$ for arbitrary $\epsilon > 0$
$\exists\, f \in \mathsf{EXP}^{\mathcal{A}} : C_f^B(n) = 2^{n^{\Omega(1)}}$	$\mathsf{EXP}^{\mathcal{A}}$	$(\log n)^{O(1)}$
$\exists\, f \in \mathsf{E}^{\mathcal{A}} : C_f^B(n) = 2^{\Omega(n)}$	$\mathsf{E}^{\mathcal{A}}$	$O(\log n)$

Table 2: Overview of pseudo-random generator constructions.

In this paper, we illustrate the power of our generalization by applying it to the following randomized processes from different areas of theoretical computer science:

- The Valiant-Vazirani random hashing procedure which prunes the number of satisfying assignments of a propositional formula to one [VV86].
- Exact learning of circuits using equivalence queries and access to an NP oracle [BCG+96].
- The construction of matrices with high rigidity [Val77].
- The construction of polynomial-size universal traversal sequences [AKL+79].

Consider the first item above. Valiant and Vazirani [VV86] give a randomized algorithm that takes as input a propositional formula ϕ, and outputs with high probability a list of formulas with the following property: If ϕ is satisfiable then some formula in the output list has exactly one satisfying assignment; if ϕ is unsatisfiable then no formula in the list is satisfiable. Strictly speaking the Valiant-Vazirani procedure is not a "BPP algorithm," and it is not clear how one could use a pseudo-random generator for BPP to derandomize it. One reason for this is that checking if the Valiant-Vazirani procedure was "successful," (i.e., it output a list of formulas with the above property) does not seem to be computable in polynomial time. Checking if the Valiant-Vazirani procedure succeeds for a particular input and choice of random bits *is* polynomial-time computable, however, if we are given access to an oracle for SAT. From this observation we can prove that the existence of a Boolean function in exponential time with high SAT-oracle circuit complexity implies the existence of a pseudo-random generator which can be used to derandomize the Valiant-Vazirani procedure.

More specifically, we show that given a propositional formula ϕ, we can construct in subexponential time a collection of polynomial-size formulas satisfying the above Valiant-Vazirani property unless the polynomial-time hierarchy collapses. If there exists a language in E with SAT-oracle circuit complexity $2^{\epsilon n}$ for some $\epsilon > 0$ we achieve a polynomial time deterministic procedure. It follows that, under the same hypothesis, we can find in polynomial time a satisfying assignment for a propositional formula using *nonadaptive* access to an oracle for SAT. The latter contrasts the standard adaptive method of binary search. We obtain derandomization results of a similar kind for the other items in the above list. We refer to Section 5 for the precise statements.

1.4 Relationship to Extractors

Our generalized hardness versus randomness trade-offs have also played a crucial role in the recent breakthrough extractor constructions by Trevisan [Tre99]. An extractor is a combinatorial object used to "extract" randomness from a corrupt or weakly random source. More specifically, applying an extractor to a probability distribution with some crude randomness results in a distribution which is almost uniform over strings of length related to the amount of crude randomness in the original distribution. We refer to the survey paper by Nisan and Ta-Shma [NT98] for background, precise definitions, and constructions.

In retrospect, from the observation that the construction due to Impagliazzo and Wigderson [IW97] is a "black-box" transformation of a hard function, one can directly prove the existence of the type of extractors found in the aforementioned work of Trevisan [Tre99]. See Section 6 for a discussion.

1.5 Universal Traversal Sequences and Randomized Logspace

We also obtain some results related to the construction of universal traversal sequences and the derandomization of randomized logspace. Universal traversal sequences are instructions for walks on arbitrary undirected graphs of a given size that have the property that any vertex of the graphs is visited at least once by the walk. The instructions only depend on the number of vertices of the graph, and not on its structure. A major open problem is to give a deterministic polynomial time (or logspace) construction of such sequences. A logspace construction of universal traversal sequences, for example, would show that undirected connectivity is computable in logspace.

We give polynomial-time constructions of universal traversal sequences under the assumption that there exists a language in E with circuit complexity $2^{\epsilon n}$ for some ϵ. We also show that if there is a language in linear space that requires circuits of size $2^{\Omega(n)}$, then $\mathsf{BPL} = \mathsf{L}$, where BPL denotes the languages recognizable in randomized logspace with bounded two-sided error. In fact, we only need the weaker hypothesis that there exists a language in linear space that requires branching programs of size $2^{\epsilon n}$. This answers a question raised by Clementi, Rolim, and Trevisan [CRT98]. As a corollary, under the same hypothesis, we can generate universal traversal sequences in logspace.

1.6 Organization

Section 2 introduces our notation. In Section 3 we generalize the techniques used to derandomize BPP and show how to use them in the Arthur-Merlin setting. Section 4 defines a broad class of randomized processes and shows how our approach allows us to reduce the randomness of any process that fits within this class. In Section 5 we apply this framework to the four examples mentioned above. We discuss directions for further research and some recent progress in Section 6.

2 Notation

Most of our complexity theoretic notation is standard. We refer the reader to the textbooks by Balcázar, Díaz and Gabarró [BDG95], and by Papadimitriou [Pap94].

An *oracle circuit* D is a circuit with AND, OR and NOT gates as well as oracle gates, which compute membership of the string formed by their input bits to some unspecified oracle B. The function D^B the circuit computes depends on the oracle B. For fixed B, we will use the term *B-oracle circuit* to denote an oracle circuit with access to B as an oracle. In this paper, we measure the *size* of a circuit by its number of connections.

Given a Boolean function $f : \{0,1\}^* \to \{0,1\}$ and an oracle B, the *circuit complexity* $C_f^B(n)$ of f at length n relative to B is the smallest integer t such that there is a B-oracle circuit of size t that computes f on inputs of length n. The *hardness* $H_f^B(n)$ of f at length n relative to B is the largest integer t such that for any oracle circuit D of size at most t with n inputs

$$|\Pr_x[D^B(x) = f(x)] - \frac{1}{2}| < \frac{1}{t},$$

where x is uniformly distributed over $\{0,1\}^n$.

A *pseudo-random generator* G is a sequence of functions $(G_n)_{n\in\mathbb{N}}$ such that G_n maps $\{0,1\}^{s(n)}$ to $\{0,1\}^n$ for some function $s : \mathbb{N} \to \mathbb{N}$ with $s(n) < n$. The function s is called the *seed length* of G. We say that G is *computable in* $\mathcal{C}$ if the problem of deciding the i-th bit of $G_n(\sigma)$ given $\langle n, \sigma, i\rangle$ belongs to $\mathcal{C}$. Given an oracle B, G_n is said to be *secure against* B if for any oracle circuit D of size at most n

$$|\Pr_\rho[D^B(\rho) = 1] - \Pr_\sigma[D^B(G_n(\sigma)) = 1]| < \frac{1}{n}, \tag{1}$$

where ρ is uniformly distributed over $\{0,1\}^n$ and σ over $\{0,1\}^{s(n)}$. G is said to be secure against B if G_n is secure against B for almost all n.

For any function $t : \mathbb{N} \to \mathbb{N}$, $\mathsf{AM\text{-}TIME}[t(n)]$ represents the class of languages L for which there exists a deterministic Turing machine M that runs in time $O(t(n))$ on inputs of the form $\langle x, y, z\rangle$ where $x \in \{0,1\}^n$ and $y, z \in \{0,1\}^{t(n)}$, such that for any input x,

$$x \in L \quad \Rightarrow \quad \Pr_{|y|=t(n)}[\exists z \in \{0,1\}^{t(n)} : M(\langle x,y,z\rangle) = 1] \geqslant \frac{2}{3} \tag{2}$$

$$x \notin L \quad \Rightarrow \quad \Pr_{|y|=t(n)}[\exists z \in \{0,1\}^{t(n)} : M(\langle x,y,z\rangle) = 1] \leqslant \frac{1}{3}, \tag{3}$$

where $n = |x|$ and the probabilities are with respect to the uniform distribution. AM denotes $\cup_{c>0}\mathsf{AM\text{-}TIME}[n^c]$. The above definition of Arthur-Merlin games allows two-sided error. Requiring one-sided error leads to the same class modulo polynomial factors. More precisely, an Arthur-Merlin game running in time $O(t(n))$ can be transformed into an equivalent Arthur-Merlin game running in time $(t(n))^{O(1)}$ satisfying (2) with the 2/3 on the right-hand side replaced by 1 [FGM$^+$89]. In particular, one can assume without loss of generality that every language in AM has a polynomial-time one-sided error Arthur-Merlin game. We will not exploit that fact in this paper but later work [MV99] crucially relies on it.

$\mathsf{EXP} = \cup_{c>0}\mathsf{DTIME}[2^{n^c}]$ and $\mathsf{E} = \cup_{c>0}\mathsf{DTIME}[2^{cn}]$. Similarly, $\mathsf{NEXP} = \cup_{c>0}\mathsf{NTIME}[2^{n^c}]$ and $\mathsf{NE} = \cup_{c>0}\mathsf{NTIME}[2^{cn}]$.

For any complexity class $\mathcal{C}$, the class i.o.-$\mathcal{C}$ consists of all languages L for which there is a language $L' \in \mathcal{C}$ such that $L \cap \{0,1\}^n = L' \cap \{0,1\}^n$ for infinitely many lengths n.

For any function $s : \mathbb{N} \to \mathbb{N}$, $\Omega(s)$ denotes the class of all functions $t : \mathbb{N} \to \mathbb{N}$ for which there exists a constant $\epsilon > 0$ such that $t(n) \geqslant \epsilon \cdot s(n)$ for almost all n. $\omega(s)$ is the class of t such that for any $c > 0$ and almost all n, $t(n) \geqslant c \cdot s(n)$.

3 Derandomizing Arthur-Merlin Games

In this section, we develop methods for derandomizing Arthur-Merlin games and give evidence that the class AM of languages with bounded round Arthur-Merlin games is not much larger than NP and may even coincide with it. For clarity of exposition and to provide an initial outline for our arguments, we first state all of our theorems without proofs. We then fill in details starting in Section 3.1.

As is customary in the area of derandomization, our approach will be to construct pseudo-random generators with appropriate security properties. The following lemma states that to derandomize $\mathsf{AM} = \cup_{c>0}\mathsf{AM\text{-}TIME}[n^c]$, the pseudo-random generator need only be secure against SAT. See Section 3.1 for a proof.

Lemma 3.1 *Let $s, t, \tau : \mathbb{N} \to \mathbb{N}$ be constructible functions. If there is a pseudo-random generator G computable in $\mathsf{NTIME}[\tau(n)] \cap \mathsf{coNTIME}[\tau(n)]$ and with seed length s that is secure against* SAT*, then*

$$\mathsf{AM\text{-}TIME}[t(n)] \subseteq \mathsf{NTIME}[2^{s(t'(n))} \cdot \tau(s(t'(n))) \cdot t(n)], \tag{4}$$

where $t'(n) = O(t(n)\log^2 t(n))$.

In order to build such a pseudo-random generator, we will extend previous work [NW94, BFNW93, IW97, STV00] to the nondeterministic setting of Arthur-Merlin games. The main construction in these papers is a reduction from a circuit that distinguishes the output of a pseudo-random generator based on f from the uniform distribution to a circuit capable of computing f. We argue that this construction works for any nonuniform model of computation satisfying certain closure properties, and in particular for B-oracle circuits for any fixed oracle B. In this way, we obtain pseudo-random generators secure against B from functions which small B-oracle circuits cannot compute.

We begin with a generalized version of the main result of Nisan and Wigderson [NW94]. Their construction works for any nonuniform model which is closed under precomputation and complementation. In particular, it carries through for oracle circuits. See Section 3.2 for a more detailed argument.

Theorem 3.2 *Let B be any oracle, g a Boolean function and $\ell : \mathbb{N} \to \mathbb{N}$ a constructible function. If*

$$H_g^B(\ell(n)) \geqslant n^2$$

then there exists a pseudo-random generator G which is secure against B and has a constructible seed length $s(n) = O(\ell^2(n)/\log n)$. Computing any bit of $G(\sigma)$ for $\sigma \in \{0,1\}^{s(n)}$ takes time $2^{O(s(n))}$ plus one evaluation of g on an input of length $\ell(n)$.

Theorem 3.2 states that the existence of functions with high average-case oracle circuit complexity, say SAT oracles, implies the existence of pseudo-random generators secure against the corresponding oracle, in this case SAT. We would like to relax this assumption from average-case to worst-case, however, so we will need to examine worst-case to average-case reductions, also known as *hardness amplifications*.

The worst-case to average-case hardness transformation of Babai et al. [BFNW93] only requires closure of the model of computation under majority, complementation, and certain arithmetic field operations. Since oracle circuits have these closure properties, the transformation also works for this model. This observation enables us to establish the first two lines of derandomization results for Arthur-Merlin games in Table 1 but falls short in establishing the last line. For the last line, sometimes referred to as a "high-end" derandomization, we appeal to the reduction in Impagliazzo and Wigderson [IW97] which also carries through for oracle circuits and is powerful enough to establish all of Table 1. An alternate to the Impagliazzo-Wigderson construction is described by Sudan et al. [STV00] which is more amenable to generalization. In Section 3.3 we provide the details of this generalization and obtain the following hardness amplification result.

Theorem 3.3 *There exist positive constants γ and δ such that for any oracle B, Boolean function f, and constructible function $h : \mathbb{N} \to \mathbb{N}$ satisfying*

$$h(n) \leqslant (C_f^B(\gamma n))^\delta / n$$

the following holds. There exists a Boolean function g such that $H_g^B(n) \geqslant h(n)$. Computing g on an input of length n takes time $2^{O(n)}$ plus evaluating f on all inputs of length γn.

Combining Theorems 3.2 and 3.3 yields our main derandomization tool. Some instantiations are given in Table 2.

Theorem 3.4 *There exists a positive constant c such that the following holds for any class $\mathcal{A}$ of oracles, oracle B, Boolean function $f \in \mathsf{E}^{\mathcal{A}}$, and constructible function $\ell : \mathbb{N} \to \mathbb{N}$. If*

$$C_f^B(\ell(n)) \geqslant n^c$$

then there exists a pseudo-random generator G computable in $\mathsf{E}^{\mathcal{A}}$ which is secure against B and has a constructible seed length $s(n) = O(\ell^2(n)/\log n)$. The same holds if E is replaced by EXP.

In this section, we will apply Theorem 3.4 with $\mathcal{A} = \mathsf{NP} \cap \mathsf{coNP}$ and $B = \mathrm{SAT}$. Note that $\mathsf{EXP}^{\mathsf{NP}\cap\mathsf{coNP}} = \mathsf{NEXP} \cap \mathsf{coNEXP}$. Together with Lemma 3.1, Theorem 3.4 yields the following result.

Theorem 3.5 *If there is a Boolean function $f \in \mathsf{NEXP}\cap\mathsf{coNEXP}$ such that $C_f^{\mathrm{SAT}}(n) = n^{\omega(1)}$, then*

$$\mathsf{AM} \subseteq \cap_{\epsilon>0}\mathsf{NTIME}[2^{n^\epsilon}].$$

We can rephrase the weak version of Theorem 3.5 as:

Theorem 3.6 *If $\mathsf{NEXP} \cap \mathsf{coNEXP} \not\subseteq \mathsf{P}^{\mathsf{NP}}/\mathsf{poly}$, then $\mathsf{AM} \subseteq \cap_{\epsilon>0}\text{i.o.-}\mathsf{NTIME}[2^{n^\epsilon}]$.*

So, if the conclusion of Theorem 3.6 fails to hold, then $\mathsf{NEXP}\cap\mathsf{coNEXP} \subseteq \mathsf{P}^{\mathsf{NP}}/\mathsf{poly}$, which implies that $\mathsf{EXP} = \Sigma_3^{\mathsf{p}} \cap \Pi_3^{\mathsf{p}}$ [KL82]. Therefore, we obtain:

Theorem 3.7 *If exponential time does not collapse to the third level of the polynomial-time hierarchy, then every language in AM, and graph nonisomorphism in particular, has subexponential size proofs for infinitely many lengths.*

More generally, Theorem 3.4 yields the following derandomization result for Arthur-Merlin games. Note that $\mathsf{E}^{\mathsf{NP}\cap\mathsf{coNP}} = \mathsf{NE} \cap \mathsf{coNE}$.

Theorem 3.8 *If there is a Boolean function $f \in \mathsf{NE}\cap\mathsf{coNE}$, and a constructible function $\ell : \mathbb{N} \to \mathbb{N}$ such that*

$$C_f^{\mathrm{SAT}}(\ell(n)) = \Omega(n),$$

then

$$\mathsf{AM} \subseteq \cup_{c>0}\mathsf{NTIME}[2^{\ell^2(n^c)/\log n}].$$

Theorem 3.8 yields a range of hardness versus randomness trade-offs for the various choices of the parameter ℓ. Since $C_f^B(n)$ is always at most $O(2^n/n)$, the hypothesis of Theorem 3.8 cannot be met for ℓ sublogarithmic. On the other hand, the conclusion becomes trivial in the case where ℓ is polynomial. Therefore, the interesting range for ℓ lies between logarithmic and subpolynomial. For example, for ℓ polylogarithmic, Theorem 3.8 (combined with some padding) yields the second line in Table 1 and the following. Recall that quasipolynomial means $2^{\log^{O(1)} n}$.

Theorem 3.9 *If there is a Boolean function $f \in \mathsf{NEXP}\cap\mathsf{coNEXP}$ such that $C_f^{\mathrm{SAT}}(n) = \Omega(2^{n^\epsilon})$ for some $\epsilon > 0$, then every language in AM has quasipolynomial size proofs.*

Theorem 3.8 achieves complete derandomizations of Arthur-Merlin games in case of logarithmic ℓ:

Theorem 3.10 *If there is a Boolean function $f \in \mathsf{NE}\cap\mathsf{coNE}$ such that $C_f^{\mathrm{SAT}}(n) \in 2^{\Omega(n)}$, then $\mathsf{AM} = \mathsf{NP}$. In particular, the same hypothesis implies that graph nonisomorphism has polynomial size proofs.*

We wish to point out that, under the hypothesis of Theorem 3.10, the proof of Lemma 3.1 describes an explicit certificate for graph nonisomorphism or any other AM-language.

We also note that for derandomizing AM it is actually sufficient to construct efficient pseudo-random generators that are secure against SAT-oracle circuits with *parallel* access to the oracle. All theorems in this section also hold for oracle circuits with such restricted access.

Finally, we note that our derandomizations also apply to Arthur-Merlin games where the number of rounds of interaction is not necessarily bounded by a constant. Via standard transformations (see [BM88]), one can show that any language possessing an Arthur-Merlin game with ℓ rounds and verifier complexity $t(n)$ is contained in $\mathsf{AM\text{-}TIME}[t'(n)]$ where $t'(n) = t(n)^{O(\ell)}$), so we can apply the theorems from this section directly.

3.1 Proof of Lemma 3.1

Let L be a language satisfying (2) and (3), and consider

$$L' = \{\langle x, y\rangle \mid y \in \{0,1\}^{t(|x|)} \text{ and } \exists\, z \in \{0,1\}^{t(|x|)} : M(\langle x, y, z\rangle) = 1\}.$$

By Cook's Theorem [Coo71, Coo88], since L' is in nondeterministic linear time, there exists a circuit of size $t'(n) = O(t(n) \log^2 t(n))$ that many-one reduces L' to SAT on inputs $\langle x, y\rangle$ with $x \in \{0,1\}^n$ and $y \in \{0,1\}^{t(n)}$. For any fixed value of $x \in \{0,1\}^n$, let C_x denote the corresponding oracle circuit obtained by hardwiring x. Note that C_x makes a single oracle query and outputs the answer to that query. Since G is a pseudo-random generator secure against SAT, conditions (2) and (3) imply that

$$\Pr_{\sigma}[C_x^{\mathrm{SAT}}(G_{t'(n)}(\sigma)) = 1] \begin{array}{ll} > \frac{1}{2} & \text{if } x \in L \\ < \frac{1}{2} & \text{if } x \notin L, \end{array} \tag{5}$$

where the probability is with respect to the uniform distribution of σ over $\{0,1\}^{s(t'(n))}$.

Since G is computable in $\mathsf{NTIME}[\tau(n)] \cap \mathsf{coNTIME}[\tau(n)]$, there exists a nondeterministic Turing machine N running in time τ that accepts $\{\langle n, \sigma, i, b\rangle \mid i\text{-th bit of } G_n(\sigma) \text{ equals } b\}$. Now, consider the nondeterministic algorithm in Figure 1.

```
counter ← 0
for every σ ∈ {0,1}^{s(t'(n))}
    for i ← 1, ..., t(n)
        guess ρ_i ∈ {0,1}
        guess a computation path p for N(⟨n, σ, i, ρ_i⟩)
        if N(⟨n, σ, i, ρ_i⟩) rejects along p
          then reject and abort
    guess z ∈ {0,1}^{t(n)}
    counter ← counter + M(⟨x, ρ, z⟩)
if counter > 2^{t'(n)−1} then accept
```

Figure 1: Nondeterministic algorithm for deciding L

Figure 1 describes a nondeterministic Turing machine that runs in time $2^{s(t'(n))} \cdot \tau(s(t'(n))) \cdot t(n)$. The largest possible value of counter at the end of the outer loop over all possible nondeterministic choices in the algorithm of Figure 1, equals $2^{s(t'(n))} \cdot \Pr_\sigma[C_x^{\mathrm{SAT}}(G_{t'(n)}(\sigma)) = 1]$. It follows from (5) that the machine accepts L. This finishes the proof of Lemma 3.1.

3.2 Proof of Theorem 3.2

This and the next section establish our generalization of known hardness versus randomness tradeoffs. The key observation is that, using the appropriate notion of relativization, the known hardness versus randomness tradeoffs *relativize*.

Relativization is the idea of giving all computing devices access to one or more oracles for arbitrary but fixed computational problems. Most theorems in computational complexity relativize, i.e., they hold under relativization. This is often just stated as a remark, as it is straightforward to verify that the proof of the theorem carries through when oracles are introduced. The few known nonrelativizing results are almost all in the area of interactive proofs and probabilistically checkable proofs. One of their distinguishing features is that they rely on considerable combinatorics and algebra. The latter are also crucial ingredients in the known hardness versus randomness trade-offs, which might lead one to believe that these trade-offs also do not relativize. However, a careful examination in this and the next section will reveal that the hardness versus randomness trade-offs do relativize. In fact, they relativize in the very strong way stated in Theorem 3.4.

We start with the construction of a pseudo-random generator out of a Boolean function that is hard on average. Nisan and Wigderson use the following notion of a combinatorial design in their construction of a pseudo-random generator.

Definition 3.11 ([NW94]) *An (m, ℓ) design of size k over a universe Ω is a collection $\mathcal{S} = (S_1, S_2, \ldots, S_k)$ of subsets of Ω, each of size ℓ, such that for any $1 \leqslant i < j \leqslant k$, the intersection $S_i \cap S_j$ has size at most m.*

In particular, in order to construct G_n, Nisan and Wigderson need a $(\log n, \ell)$ design of size n for a given value of $\ell \geqslant \log n$. The size of the design universe will be the seed length $s(n)$ of G_n and therefore should be small. Nisan and Wigderson [NW94] show that the greedy approach works and that it yields a $(\log n, \ell)$ design of size n over a universe of size $O(\ell^2/\log n)$. See [Tre99, Lemma 7] for a detailed analysis.

Lemma 3.12 ([NW94]) *A $(\log n, \ell)$ design of size n can be constructed over a universe of size $s \in O(\ell^2/\log n)$ in time $2^s \cdot n^{O(1)}$.*

Given a function $g : \{0,1\}^\ell \to \{0,1\}$ and a $(\log n, \ell)$ design $\mathcal{S} = (S_1, S_2, \ldots, S_n)$ over $\{1, 2, \ldots, s\}$, Nisan and Wigderson define their pseudo-random generator G_n as follows:

$$G_n : \{0,1\}^s \to \{0,1\}^n : \sigma \to g(\sigma|_{S_1})\, g(\sigma|_{S_2}) \ldots g(\sigma|_{S_n}), \tag{6}$$

where $\sigma|_{S_i}$ denotes the substring of σ consisting of the bits indexed by the elements from S_i.

The following theorem, combined with Lemma 3.12, finishes the proof of Theorem 3.2.

Theorem 3.13 *Let B be any oracle, $g : \{0,1\}^\ell \to \{0,1\}$, and $\mathcal{S} = (S_1, S_2, \ldots, S_n)$ a $(\log n, \ell)$ design over $\{1, 2, \ldots, s\}$. If $H_g^B \geqslant n^2$ then G_n as defined by (6) is secure against B.*

The proof goes along the lines of the one by Nisan and Wigderson [NW94]. It is an application of Yao's observation that distinguishability from the uniform distribution implies predictability [Yao82], and uses the so-called hybrid argument [GM84].

Proof:
We proceed by contradiction. Assume that G_n is not secure against B. This means that there exists an oracle circuit D of size at most n such that either

$$\Pr_\sigma[D^B(G_n(\sigma)) = 1] - \Pr_\rho[D^B(\rho) = 1] > \frac{1}{n}, \tag{7}$$

or else $\Pr_\rho[D^B(\rho) = 1] - \Pr_\sigma[D^B(G_n(\sigma)) = 1] > \frac{1}{n}$, where ρ is uniformly distributed over $\{0,1\}^n$ and σ over $\{0,1\}^s$. We will assume that (7) holds. The other case is similar.

Consider the following sequence of "hybrid" distributions $\mathcal{D}_0, \ldots, \mathcal{D}_n$:

$$
\begin{array}{rcccccccccc}
\mathcal{D}_0 & = & g(\sigma|_{S_1}) & g(\sigma|_{S_2}) & & & \cdots & & & g(\sigma|_{S_{n-1}}) & g(\sigma|_{S_n}) \\
\mathcal{D}_1 & = & \rho_1 & g(\sigma|_{S_2}) & & & \cdots & & & g(\sigma|_{S_{n-1}}) & g(\sigma|_{S_n}) \\
\vdots & & & & & & \vdots & & & & \\
\mathcal{D}_i & = & \rho_1 & \rho_2 & \cdots & \rho_{i-1} & \rho_i & g(\sigma|_{S_{i+1}}) & \cdots & g(\sigma|_{S_{n-1}}) & g(\sigma|_{S_n}) \\
\vdots & & & & & & \vdots & & & & \\
\mathcal{D}_{n-1} & = & \rho_1 & \rho_2 & & & \cdots & & & \rho_{n-1} & g(\sigma|_{S_n}) \\
\mathcal{D}_n & = & \rho_1 & \rho_2 & & & \cdots & & & \rho_{n-1} & \rho_n
\end{array}
$$

As before, ρ is uniformly distributed over $\{0,1\}^n$ and σ over $\{0,1\}^s$.

Since the left-hand side of (7) can be written as

$$\Pr_{y \in \mathcal{D}_0}[D^B(y)=1] - \Pr_{z \in \mathcal{D}_n}[D^B(z)=1] = \sum_{i=1}^{n} \Pr_{y \in \mathcal{D}_{i-1}}[D^B(y)=1] - \Pr_{z \in \mathcal{D}_i}[D^B(z)=1],$$

there must exist an index i, $1 \leqslant i \leqslant n$, such that

$$\Pr_{y \in \mathcal{D}_{i-1}}[D^B(y)=1] - \Pr_{z \in \mathcal{D}_i}[D^B(z)=1] \geqslant \frac{1}{n^2}. \tag{8}$$

The only difference between $\mathcal{D}_{i-1}$ and $\mathcal{D}_i$ is the way the ith bit is generated: In $\mathcal{D}_{i-1}$ it is set to $g(\sigma|_{S_i})$ whereas in $\mathcal{D}_i$ it is a random bit. Equation (8) says that in the former case the B-oracle circuit D^B is more likely to accept than in the latter case. This suggests the following randomized predictor $P(x)$ for $g(x)$:

- Pick $\rho_1, \rho_2, \ldots, \rho_i \in \{0,1\}$ uniformly at random.
- Set $\sigma|_{S_i}$ equal to x and pick the other bits of σ uniformly at random.
- If $D^B(\rho_1, \rho_2, \ldots, \rho_i, g(\sigma|_{S_{i+1}}), \ldots, g(\sigma|_{S_n}))$ accepts then output ρ_i; otherwise output $\overline{\rho_i}$ (the complement of ρ_i).

Claim 3.14

$$\Pr_{\rho,\sigma}[P(x)=g(x)] - \frac{1}{2} = \Pr_{y \in \mathcal{D}_{i-1}}[D^B(y)=1] - \Pr_{z \in \mathcal{D}_i}[D^B(z)=1], \tag{9}$$

where x denotes $\sigma|_{S_i}$.

We argue the claim by conditioning on all of ρ and σ except ρ_i. Let $\tilde{y}$ denote y with the ith bit flipped.

- Case $D^B(y) = D^B(\tilde{y})$.
  Since the value of the ith component doesn't affect D^B, the right-hand side of (9) vanishes. $P(x)$ is a random bit in this case so the left-hand side also vanishes.
- Case $D^B(y) \neq D^B(\tilde{y})$.
  Then $D^B(z)$ accepts with probability $\frac{1}{2}$, and $P(x)$ equals $g(x)$ precisely when $D^B(y)=1$.

This finishes the proof of the claim.

Note that $x \doteq \sigma|_{S_i}$ is uniformly distributed over $\{0,1\}^m$. By an averaging argument, (8) and (9) imply that we can fix ρ and all of σ outside of S_i such that

$$\Pr_x[P(x)=g(x)] \geqslant \frac{1}{2} + \frac{1}{n^2}. \tag{10}$$

Note that

$$P(x) = D^B(\rho_1, \rho_2, \dots, \rho_i, g(\sigma|_{S_{i+1}}), \dots, g(\sigma|_{S_n})) \oplus \overline{\rho_i}. \tag{11}$$

Since $|S_j \cap S_i| \leqslant \log n$ for $j \neq i$, each of the components $g(\sigma|_{S_j})$, $i < j \leqslant n$, only depend on $\log n$ bits of x and therefore can be computed by circuits of size at most n with input x. There are at most $n-1$ such components. As ρ is fixed and the size of D if at most n, it follows that the right-hand side of (11) can be evaluated by a B-oracle circuit C^B of size at most $(n-1)n + n = n^2$. In combination with (10) this leads to the contradiction that $H_g^B < n^2$. □

3.3 Proof of Theorem 3.3

Sudan et al. showed the following list decoding result which is used to give a strong worst-case to average-case hardness amplification [STV00, Definition 22 and Lemma 28]:

Theorem 3.15 ([STV00]) *There exists a family of maps $(\mathcal{C}_{N,\epsilon})_{N \in \mathbb{N}, \epsilon > 0}$ where $\mathcal{C}_{N,\epsilon} : \{0,1\}^N \to \{0,1\}^{N'(N,\epsilon)}$ such that the following hold.*

- *$\mathcal{C}_{N,\epsilon}$ is computable in time polynomial in N and $1/\epsilon$. In particular, $N' \doteq N'(N,\epsilon)$ is polynomial in N and $1/\epsilon$.*
- *There exists a randomized algorithm that, on input N and ϵ and given oracle access to a string $y \in \{0,1\}^{N'}$, runs in time polynomial in $\log N$ and $1/\epsilon$ and outputs with high probability a list of randomized oracle machines M_j with the next properties:*
  - *For every $x \in \{0,1\}^N$ such that $(\mathcal{C}_{N,\epsilon}(x))_i = y_i$ for at least $(\frac{1}{2} + \epsilon)N'$ of the positions i, $1 \leqslant i \leqslant N'$, there exists an index j such that for any position i, $1 \leqslant i \leqslant N$, $M_j^y(i)$ outputs x_i with high probability.*
  - *Each machine M_j runs in time polynomial in $\log N$ and $1/\epsilon$.*

By Adleman's argument [Adl78] we can transform the randomized procedures in the second bullet of Theorem 3.15 into circuits. This yields the following corollary.

Corollary 3.16 *There exists a constant c such that for any $N \in \mathbb{N}$, $x \in \{0,1\}^N$ and $y \in \{0,1\}^{N'}$ satisfying $(\mathcal{C}_{N,\epsilon}(x))_i = y_i$ for at least $(\frac{1}{2} + \epsilon)N'$ of the positions i, $1 \leqslant i \leqslant N'$, there exists an oracle circuit D of size $(\frac{1}{\epsilon}\log N)^c$ such that $D^y(i) = x_i$ for every $1 \leqslant i \leqslant N$.*

We define g such that its truth-table at length $n' \doteq \log N'$ equals the encoding $\mathcal{C}_{N,\epsilon}$ of the truth-table of f at length $n \doteq \log N$. We set $\epsilon = 1/h(n')$ where $h(n')$ is the hardness level we want to achieve. Since $h(n') \in O(2^{n'}/n')$, the first bullet in Theorem 3.15 implies that N and N' are polynomially related, so we can set $n = \gamma \cdot n'$ for some positive constant γ.

Now, suppose that there exists a B-oracle circuit of size at most $h(n')$ that computes g correctly on at least $(\frac{1}{2} + \frac{1}{h(n')})N'$ of the inputs of length n'. By plugging in this circuit as the oracle y in Corollary 3.16, we obtain a B-oracle circuit that computes f at length $n = \gamma \cdot n'$ correctly and has size at most $(\frac{1}{\epsilon}\log N)^c \cdot h(n') = (\gamma n')^c \cdot (h(n'))^{c+1}$. In other words,

$$C_f^B(\gamma n') \leqslant (\gamma n')^c \cdot (h(n'))^{c+1}. \tag{12}$$

For $\delta < 1/(c+1)$, (12) contradicts the upper bound on $h(n')$ in the hypothesis of Theorem 3.3. Therefore, $H_g^B(n') \geqslant h(n')$. □

4 A General Framework for Derandomization

In the previous section, we showed that an Arthur-Merlin protocol can be viewed as a SAT-oracle distinguisher for a pseudo-random generator, and if a Boolean function f exists with sufficient hardness against SAT-oracle circuits we can construct a pseudo-random generator based on f that will look random to our Arthur-Merlin protocol. Still, we have only applied our results to randomized *decision* algorithms. In this section, we show how to relax this condition and obtain hardness versus randomness trade-offs for a broader class of randomized processes. Under a sufficient hardness condition depending upon the particular randomized algorithm, we are able to reduce the algorithm's randomness to a logarithmic factor and, in some cases, provide a complete derandomization. Weaker hardness conditions yield partial derandomizations.

We first define the notion of a *randomized process* to which our approach applies:

Definition 4.1 *A randomized process that uses $r(n)$ random bits on inputs of length n is a pair (F, π) where:*

- *F is a function that takes a string x and a string ρ of length $r(|x|)$, and outputs the outcome of the process on input x using ρ as the random bit sequence.*
- *π is a predicate with the same domain as F. Intuitively π indicates whether the process succeeds on input x using ρ, i.e., whether ρ is a "good" choice of random bits for the given input x. We call π the success predicate of the randomized process.*

In the case of Arthur-Merlin games F is Boolean. More specifically, for the game defined by (2) and (3) in Section 2, $F(x, \rho)$ coincides with the predicate $\exists z \in \{0,1\}^{t(n)} : M(\langle x, \rho, z\rangle) = 1$; the success predicate $\pi(x, \rho)$ equals $F(x, \rho)$ if $x \in L$ and the complement of $F(x, \rho)$ otherwise. In the specific randomized processes we will consider in Section 5, F will be non-Boolean.

What matters for our derandomization results is the complexity of the success predicate π and, more precisely, the property stated in the following straightforward lemma:

Lemma 4.2 *Let B be any oracle, and (F, π) a randomized process using $r(n)$ random bits such that for any fixed input x of length n, the predicate $\pi_x : \{0,1\}^{r(n)} \to \{0,1\}$ where $\pi_x(\rho) \doteq \pi(x, \rho)$ can be decided by a B-oracle circuit of size $t(n) \geqslant r(n)$. If G is a pseudo-random generator with seed length $s(n)$ which is secure against B, then for any input x of length n*

$$|\Pr_{\rho}[\pi(x, \rho) = 1] - \Pr_{\sigma}[\pi(x, G_{t(n)}(\sigma)[1..r(n)]) = 1]| = O(1/t(n)),$$

where ρ is uniformly distributed over $\{0,1\}^{r(n)}$ and σ over $\{0,1\}^{s(t(n))}$.

In our applications we will be concerned with randomized processes that use a polynomial number of random bits. We will choose the oracle B in Lemma 4.2 powerful enough so that it can check the process efficiently in the following sense.

Definition 4.3 *Let B be an oracle, and (F, π) a randomized process using a polynomial number $r(n)$ of random bits. We say that B can* efficiently check *(F, π) if there is a polynomial p such that for any fixed input x of length n, the predicate $\pi_x : \{0,1\}^{r(n)} \to \{0,1\}$ where $\pi_x(\rho) \doteq \pi(x, \rho)$ can be decided by a B-oracle circuit of size $p(n)$.*

Using the success predicate as a distinguisher as in Lemma 4.2, Theorem 3.4 yields our general derandomization result.

Theorem 4.4 *Let $\mathcal{A}$ be a class of oracles, B an oracle, d a positive constant, and $\ell : \mathbb{N} \to \mathbb{N}$ a constructible function. Let (F, π) be a randomized process using a polynomial number of random bits such that B can efficiently check (F, π). If there exists a Boolean function $f \in \mathsf{E}^{\mathcal{A}}$ such that $C_f^B(\ell(n)) = \Omega(n)$, then there exists a function G computable in $\mathsf{E}^{\mathcal{A}}$ and a constructible function $s(n) = O(\ell^2(n^{O(1)})/\log n)$ such that for any input x of length n*

$$|\Pr_{\rho}[\pi(x, \rho) = 1] - \Pr_{\sigma}[\pi(x, G(\sigma)) = 1]| = O(1/n^d),$$

where ρ is uniformly distributed over $\{0,1\}^{r(n)}$ and σ over $\{0,1\}^{s(n)}$. The same holds if E is replaced by EXP.

See Table 2 for some interesting instantiations of Theorem 4.4.

In order to reduce the randomness of a randomized process, we will first analyze the complexity of an oracle B capable of efficiently checking the associated success predicate and then construct a pseudo-random generator secure against B based on a function with presumed hardness against B. We will see several examples of this in the next section.

5 More Applications

We will now apply the general framework of Section 4 to various other constructions in computational complexity. As is customary, we only state our results in terms of the strongest of the assumptions in Table 2, yielding polynomial time deterministic simulations. It should be noted, however, that weaker assumptions can be taken (e.g., that the polynomial-time hierarchy does not collapse) in order to achieve weaker, but still subexponential, deterministic simulations.

5.1 Valiant-Vazirani

Our first example is the randomized Boolean hashing protocol developed by Valiant and Vazirani [VV86]. They give a method for pruning the satisfying assignments of a propositional formula to one.

Theorem 5.1 ([VV86]) *There exists a randomized polynomial time algorithm that, on input a propositional formula ϕ, outputs a list of propositional formulas such that:*

- *Every satisfying assignment to any of the formulas in the list also satisfies ϕ.*
- *If ϕ is satisfiable, then with high probability at least one of the formulas in the list has exactly one satisfying assignment.*

Let $F(x, \rho)$ denote the list of formulas the Valiant-Vazirani algorithm produces on input x using coin flips specified by ρ. We define the success predicate $\pi(x, \rho)$ to hold unless x is a satisfiable formula and none of the formulas in $F(x, \rho)$ has a unique satisfying assignment. It is clear that (F, π) corresponds to a formalization of the Valiant-Vazirani process and fits within our framework.

Theorem 5.2 *If there is a Boolean function $f \in \mathsf{E}$ such that $C_f^{\mathrm{SAT}}(n) = 2^{\Omega(n)}$, then, given a propositional formula ϕ, we can generate in polynomial time a list of propositional formulas such that:*

- *Every satisfying assignment to any of the formulas in the list also satisfies ϕ.*

- *If ϕ is satisfiable, then at least one of the formulas in the list has exactly one satisfying assignment.*

Proof:
Let (F, π) be the formalization as described above. Notice that checking whether a given propositional formula has at least two satisfying assignments is an NP question. Hence, we can check, in polynomial time, whether a propositional formula has a unique satisfying assignment using two queries to SAT. It follows that SAT can efficiently check (F, π). Applying theorem 4.4 to (F, π) yields a pseudo-random generator G computable in E that looks random to the Valiant-Vazirani process. Enumerating over all seeds and collecting all formulas produces the desired list of formulas in polynomial time. □

We now have the following corollary about computing satisfying assignments nonadaptively:

Corollary 5.3 *If there is a Boolean function $f \in \mathsf{E}$ such that $C_f^{\mathrm{SAT}}(n) = 2^{\Omega(n)}$, then, given a satisfiable propositional formula ϕ, we can find a satisfying assignment for ϕ in polynomial time given nonadaptive oracle access to* SAT*.*

Proof:
A satisfying assignment to a uniquely satisfiable propositional formula ψ can be found in polynomial time using nonadaptive oracle queries to SAT: We set a variable v of ψ iff the formula obtained from ψ by substituting TRUE for v is satisfiable.

We run the latter procedure on each formula ψ of the list produced by Theorem 5.2 on input ϕ, and output the first satisfying assignment we obtain. □

We also obtain interesting structural observations. Recall that the class $\#\mathsf{P}$ contains all functions $f : \Sigma^* \to \mathbb{N}$ for which there exists a nondeterministic polynomial-time Turing machine M such that $f(x)$ equals the number of accepting computations of M on input x. A language L belongs to $\oplus\mathsf{P}$ if there exists a $\#\mathsf{P}$ function f such that an input x belongs to L iff $f(x)$ is odd. A GapP function is the difference of two $\#\mathsf{P}$ functions, and SPP denotes the class of all languages whose characteristic function is a GapP function. The class PP contains all languages L for which there exists an $f \in \mathsf{GapP}$ such that $x \in L$ iff $f(x) > 0$. SPP is contained in both $\oplus\mathsf{P}$ and PP. We refer the reader to the survey by Fortnow [For97] for background on these counting classes. Theorem 5.2 implies:

Corollary 5.4 *If there is a Boolean function $f \in \mathsf{E}$ such that $C_f^{\mathrm{SAT}}(n) = 2^{\Omega(n)}$, then NP is contained in SPP.*

Proof:
Let $h(\phi, i)$ denote the $\#\mathsf{P}$ function that gives the number of satisfying assignments to the ith formula of the list produced by Theorem 5.2 on input ϕ. Then

$$1 - \prod_i (1 - h(\phi, i)) \tag{13}$$

equals the characteristic function of SAT. By the closure properties of GapP [FFK94], (13) belongs to GapP. □

Similarly, we can conditionally derandomize the result by Toda and Ogiwara [TO92] that the polynomial-time hierarchy does not add power to GapP in a randomized setting. Applying our techniques to their main lemma yields the following theorem:

Lemma 5.5 *Let B be any oracle. If there is a Boolean function $f \in \mathsf{E}$ such that $C_f^{\mathrm{SAT}^B}(n) = 2^{\Omega(n)}$, then $\mathsf{GapP}^{\mathsf{NP}^B}$ is contained in GapP^B.*

This allows us to show:

Theorem 5.6 *For any integer $k \geqslant 1$ the following holds. If there is a Boolean function $f \in \mathsf{E}$ such that $C_f^{\mathrm{TQBF}_k}(n) = 2^{\Omega(n)}$, then Σ_k^{p} is contained in SPP.*

Here TQBF_k denotes the language of all true fully quantified propositional formulas with at most $k-1$ quantifier alternations.

Proof:
Corollary 5.4 relativizes and, under the given assumption, implies that the characteristic function of any language in Σ_k^{p} is contained in $\mathsf{GapP}^{\mathrm{TQBF}_{k-1}}$. The successive application of Lemma 5.5 with $B = \mathrm{TQBF}_{k-2}$, $B = \mathrm{TQBF}_{k-3}$, $\dots$, $B = \mathrm{TQBF}_1$, and $B = \varnothing$ yields under the given assumption that $\mathsf{GapP}^{\mathrm{TQBF}_k}$ is contained in GapP. □

Corollary 5.7 *Let B be any oracle hard for the polynomial-time hierarchy under polynomial-time Turing reductions. If there is a Boolean function $f \in \mathsf{E}$ such that $C_f^B(n) = 2^{\Omega(n)}$, then the polynomial-time hierarchy is contained in SPP.*

Corollary 5.7 is strongly related to Toda's Theorem [Tod91] that the polynomial-time hierarchy lies in $\mathsf{BP} \cdot \oplus\mathsf{P}$. Applying our derandomization technique directly to Toda's Theorem along with the observation that $\oplus\mathsf{P}$ is closed under majority [PZ83] (in fact $\oplus\mathsf{P}^{\oplus\mathsf{P}} = \oplus\mathsf{P}$) yields the following:

Theorem 5.8 *If there is a Boolean function $f \in \mathsf{E}$ such that $C_f^{\oplus\mathrm{SAT}}(n) = 2^{\Omega(n)}$, then the polynomial-time hierarchy is contained in $\oplus\mathsf{P}$.*

5.2 Learning Circuits

Learning theory represents another area where we can apply our techniques. We will focus on exact concept learning with equivalence queries [Ang88, Lit88], in which the learner presents hypotheses to a teacher, who then tells the learner whether the hypothesis agrees with the concept in question. If it does, the learner has succeeded; otherwise the teacher provides a counterexample and the learner continues.

A fundamental question is whether we can efficiently learn Boolean circuits in this model. If $\mathsf{P} = \mathsf{NP}$, we can; if one-way functions exist, we cannot [PW90]. Without any complexity theoretic assumption, Bshouty et al. [BCG+96] showed that access to an NP oracle and to a source of randomness suffice to efficiently learn Boolean circuits. We give evidence that we may be able to dispense with the source of randomness.

Theorem 5.9 *If there is a Boolean function $f \in \mathsf{NE} \cap \mathsf{coNE}$ such that $C_f^{\mathrm{TQBF}_2}(n) = 2^{\Omega(n)}$, then we can perform the following task in deterministic polynomial time given access to an NP oracle: exactly learn Boolean circuits of size t using equivalence queries with hypotheses that are circuits of size $O(tn + n \log n)$.*

As before, TQBF_2 denotes the language of all true fully quantified propositional formulas with at most one quantifier alternation.

Proof:
We apply Theorem 4.4 to a randomized process (F, π) underlying the algorithm of Bshouty et

al. [BCG$^+$96]. We define F in the obvious way, namely as the candidate equivalent circuit the learner outputs. The specification of the success predicate π is somewhat more involved than in the examples we have seen so far. We only want to consider a random seed as "good" if the learner, when using this seed, finds an equivalent circuit no matter how the teacher picks the counterexamples. Therefore, we define $\pi(x, \rho)$ to indicate whether for every choice of candidate counterexamples, either one of them fails to be a valid counterexample on input x and random seed ρ, or else the learner ends up with an equivalent circuit to x. It follows from the construction of Bshouty et al. that π is a Π_2^{p} predicate, and their analysis shows that for any input x, $\pi(x, \rho)$ holds with high probability.

Using the derandomization provided by Theorem 4.4, we end up with a polynomial number of candidate circuits at least one of which is equivalent to x. So, we just present each of these to the teacher and will succeed. $\square$

Bshouty et al. used their learning theory result to improve the known collapse of the polynomial-time hierarchy in case NP would have polynomial size circuits: They showed that the latter implies that the polynomial-time hierarchy is contained in $\mathsf{ZPP}^{\mathsf{NP}}$. Along similar lines, we obtain the following derandomization.

Corollary 5.10 *If* NP *has polynomial size circuits and there is a Boolean function* $f \in \mathsf{E}^{\mathsf{NP}}$ *such that* $C_f(n) = 2^{\Omega(n)}$*, then the polynomial-time hierarchy is contained in* P^{NP}*.*

Proof:
If NP has polynomial-size circuits and there is a Boolean function $f \in \mathsf{E}^{\mathsf{NP}}$ such that $C_f(n) = 2^{\Omega(n)}$, then the same function f also satisfies $C_f^{\mathrm{SAT}}(n) = 2^{\Omega(n)}$. Therefore, by applying Theorem 3.4 with $\mathcal{A} = \mathsf{NP}$ and $B = \mathrm{SAT}$, we obtain a pseudo-random generator G computable in E^{NP} which is secure against SAT and has logarithmic seed length. If follows that $\mathsf{ZPP}^{\mathsf{NP}} = \mathsf{P}^{\mathsf{NP}}$. To finish the proof, it suffices to combine this with the Bshouty et al. [BCG$^+$96] result that the polynomial-time hierarchy collapses to $\mathsf{ZPP}^{\mathsf{NP}}$ if NP has polynomial-size circuits. $\square$

5.3 Rigid Matrices

Several researchers have studied the problem of finding explicit constructions of combinatorial objects that have been proven to exist non-constructively (by using the probabilistic method, for example). In many cases, an explicit construction of some combinatorial object yields an interesting complexity theoretic result. One of the notable examples of this is the problem of matrix rigidity. The rigidity of a matrix M over a ring S, denoted $R_M^S(r)$, is the minimum number of entries of M that must be changed to reduce its rank to r or below (an entry can be changed to any element of S). Valiant [Val77] proved that an explicit construction of an infinite family of highly rigid matrices yields a circuit lower bound.

Theorem 5.11 ([Val77]) *Let* $\epsilon, \delta > 0$ *be constants. For any positive integer* n*, let* M_n *be an* $n \times n$ *matrix over a ring* S_n*. If* $R_{M_n}^{S_n}(\epsilon n) \geqslant n^{1+\delta}$ *for infinitely many values of* n*, then the linear transformations defined by the family* M_n *cannot be computed by linear size, log-depth circuits consisting of gates computing binary linear operators on* S_n*.*

Valiant also proved that almost all matrices over an infinite field have rigidity $(n-r)^2$ and almost all matrices over a fixed finite field have rigidity $\Omega((n-r)^2/\log n)$. The best known explicit constructions achieve rigidity $\Omega(n^2/r)$ over infinite fields [PV91, Raz] and $\Omega(\frac{n^2}{r}\log(\frac{n}{r}))$ [Fri90] over finite fields. These are not sufficient to obtain circuit lower bounds using Theorem 5.11.

Under a hardness assumption, our derandomization technique will give an explicit construction to which Theorem 5.11 applies: a family of matrices M_n over $S_n = \mathbb{Z}_{p(n)}[x]$ such that $R^{S_n}_{M_n}(r) \in \Omega((n-r)^2/\log n)$, where $p(n)$ is polynomially bounded.

We will use the following lemma, which follows directly from Valiant's paper [Val77].

Lemma 5.12 ([Val77]) *Let S be a ring with at least 2 elements, and n a positive integer. All but at most a $\frac{1}{n}$ fraction of the $n \times n$ matrices M over S satisfy*

$$R^S_M(r) \geqslant \frac{(n-r)^2 - 2n - 2\log n}{1 + 2\log n}. \tag{14}$$

We can use our technique to achieve the nonconstructive rigidity bounds by noticing that there exist polynomial-sized SAT-oracle circuits which can check if a matrix is rigid.

Theorem 5.13 *If there exists a Boolean function $f \in \mathsf{E}$ such that $C^{\mathrm{SAT}}_f(n) = 2^{\Omega(n)}$, then given integers $n \geqslant 1$ and $p \geqslant 2$, we can construct in time polynomial in $n + \log p$ a list of $n \times n$ matrices M over $S = \mathbb{Z}_p$ most of which satisfy (14).*

Proof:
Let ρ be a string of length $n^2 \cdot \lceil \log p \rceil$. We will view ρ as the concatenation of n^2 blocks of $\lceil \log p \rceil$ bits each. Let $F(\langle 1^n, p\rangle, \rho)$ denote the matrix whose ij-th entry is the $(n(i-1)+j)$-th block of ρ interpreted as a number in binary and taken modulo p. We define the predicate $\pi(\langle 1^n, p\rangle, \rho)$ to be true if $M = F(\langle 1^n, p\rangle, \rho)$ satisfies (14). Note that the latter is a coNP predicate: If (14) is violated for some r, we can guess modified values for fewer than the right-hand side of (14) many entries of M and verify that the rank of the modified matrix over S is at most r. So, we can decide π by one query to an oracle for SAT. Moreover, Lemma 5.12 states that for most sequences ρ, the predicate holds. By applying theorem 4.4 to (F, π) we obtain a pseudo-random generator which on most seeds outputs a matrix with the required rigidity property. Enumerating over all seeds gives us the desired list of matrices in polynomial time. $\square$

Now we need to combine this list of matrices into a single matrix with similarly high rigidity. We can do so by switching to the ring of univariate polynomials over $\mathbb{Z}_p$.

Lemma 5.14 *Given $n \times n$ matrices $M_0, M_1, \ldots, M_k$ over $\mathbb{Z}_p$ where p is a prime larger than k, we can construct in time polynomial in $n + k + \log p$ an $n \times n$ matrix N over $\mathbb{Z}_p[x]$ such that*

$$R^{\mathbb{Z}_p[x]}_N(r) \geqslant \max_{0 \leqslant i \leqslant k} R^{\mathbb{Z}_p}_{M_i}(r), \tag{15}$$

where the entries of N are polynomials of degree at most k.

Proof:
Let $q_{ij}(x)$ be the polynomial of degree at most k such that $q_{ij}(\ell)$ equals the ij-th entry of M_ℓ, $0 \leqslant \ell \leqslant k$, i.e., q_{ij} interpolates the ij-th entries of all $k+1$ matrices. Note that the q_{ij}'s exist and that each coefficient can be computed in time polynomial in $k + \log p$. Let N be the matrix whose ij-th entry is q_{ij}. We now argue that (15) holds.

Let $m = R^{\mathbb{Z}_p[x]}_N(r)$, and let N' be a matrix obtained by changing m entries of N such that the rank of N' is at most r. Then every $(r+t) \times (r+t)$ minor of N' has determinant 0 for $0 < t \leqslant n-r$. Thus the determinant of any $(r+t) \times (r+t)$ minor of N' can be viewed as an identically 0 polynomial in its entries. Let $\phi_a(N')$ be the matrix obtained by substituting $a \in \mathbb{Z}_p$ for x in every entry of N' (recall N' has entries from $\mathbb{Z}_p[x]$). Every $(r+t) \times (r+t)$ minor of $\phi_a(N')$ has determinant 0 for $0 < t \leqslant n-r$. We conclude by noticing that changing m or fewer entries of $\phi_a(N)$ has reduced its rank to a value less than or equal to r. But $\phi_a(N)$ equals M_a for $a \in \{0, 1, \ldots, k\}$. Therefore, $R^{\mathbb{Z}_p}_{M_i}(r) \leqslant m$ for any $0 \leqslant i \leqslant k$, i.e., (15) holds. □

Given this construction of rigid matrices we can conclude the following relationship among circuit lower bounds:

Theorem 5.15 *If there exists a function $f \in \mathsf{E}$ such that $C^{\mathrm{SAT}}_f(n) = 2^{\Omega(n)}$ then there exists a polynomially bounded function $p(n)$ and a polynomial-time computable family of matrices M_n where M_n is an $n \times n$ matrix over $\mathbb{Z}_{p(n)}[x]$ such that the linear transformations defined by the family M_n cannot be computed by log-depth linear size circuits which have special gates that can compute binary linear operators over $\mathbb{Z}_{p(n)}[x]$.*

Proof:
For any polynomial time computable function $p(n)$, Theorem 5.13 allows us to efficiently compute a list of $(n + \log p(n))^c$ matrices M over $\mathbb{Z}_{p(n)}$ for some constant c, most of which are rigid. Provided $p(n)$ is a prime satisfying

$$p(n) \geqslant (n + \log p(n))^c, \tag{16}$$

Lemma 5.14 efficiently combines them into a single matrix N over $\mathbb{Z}_{p(n)}[x]$ which satisfies a rigidity condition sufficient for Theorem 5.11. The smallest prime value for $p(n)$ that satisfies (16) is polynomially bounded in n, and we can compute it in time polynomial in n. □

We point out that the weaker assumption that there exists a function $f \in \mathsf{E}$ such that $C_{f(n)} = 2^{\Omega(n)}$ immediately implies (by padding) the existence of a function in P with superpolynomial circuit complexity, regardless of the depth restrictions. Viewed that way, Theorem 5.15 uses a stronger hypothesis to produce a weaker conclusion, namely the existence of a function in P that requires superpolynomial size but only for circuits of restricted depth. However, Theorem 5.15 adds credibility to Valiant's program of exhibiting an explicit function that cannot be computed by linear-size log-depth circuits using matrix rigidity. Theorem 5.15 shows that under a reasonable assumption, Valiant's program will work.

5.4 Universal Traversal Sequences

Universal traversal sequences, introduced by Cook, form another example where explicit constructions have important complexity theoretic implications. A universal traversal sequence for size n is a sequence σ of labels from $\{1, 2, \ldots, n-1\}$ such that for any undirected connected graph G with n vertices in which the incident edges at every vertex have been assigned distinct labels from $\{1, 2, \ldots, n-1\}$, the following process always visits every vertex of G: Pick an arbitrary start vertex, and in subsequent steps, go along the edge with the label matching the next symbol of σ; in case of no match, stay put during that step and continue with the next symbol of σ.

If we can construct universal traversal sequences in logspace, then we can solve undirected graph connectivity in logspace, and symmetric logspace equals logspace [LP82]. However, we do not know how to generate universal traversal sequences in logspace or even in polynomial time. Aleliunas et al. [AKL+79] showed that most sequences of length $O(n^3)$ over $\{1, 2, \ldots, n-1\}$ are universal traversal sequences for size n, but as of now the best explicit construction, due to Nisan [Nis92], yields universal traversal sequences of length $n^{O(\log n)}$. We give evidence supporting the belief that explicit universal traversal sequences of polynomial size can be generated efficiently.

A straightforward application of our technique would yield a polynomial-time construction under the assumption that E requires exponential size SAT-oracle circuits. Since being a universal traversal sequence is a coNP predicate, Theorem 4.4 applied to the Aleliunas et al. process and oracle $B = \mathrm{SAT}$ would efficiently generate under that hypothesis a collection of sequences most of which are universal traversal sequences. Concatenating all of them would yield the desired universal traversal sequence of polynomial size. However, we can do better and dispense with the oracle B.

Theorem 5.16 *If there is a Boolean function $f \in E$ such that $C_f(n) = 2^{\Omega(n)}$, then we can construct universal traversal sequences in polynomial time.*

Proof:
Let G encode a graph with n vertices with edge labels as above. For any sequence σ over $\{1, 2, \ldots, n-1\}$, let $\tau(G, \sigma)$ indicate whether for every vertex v of G, the walk in G starting from v and dictated by σ visits every vertex of G. Let $F(G, \rho)$ denote the sequence of length $c \cdot n^3$ over $\{1, 2, \ldots, n-1\}$ (where c is some sufficiently large constant) as specified by the successive bits of ρ. Let $\pi(G, \rho)$ equal $\tau(G, F(G, \rho))$. Note that π is a P predicate, so $B = \varnothing$ can efficiently check the randomized process (F, π). Since every universal traversal sequence σ satisfies $\tau(G, \sigma)$, the result of Aleliunas et al. [AKL+79] shows that for any graph G, $\pi(G, \rho)$ holds for most ρ. Therefore, Theorem 4.4 allows us to generate in polynomial time a collection of sequences σ most of which satisfy $\tau(G, \sigma)$. Their concatenation forms a single sequence σ' satisfying $\tau(G, \sigma')$. Since the σ's are independent of G, so is their concatenation σ'. Hence, we have constructed a sequence σ' which satisfies $\tau(G, \sigma')$ for every edge labeled graph G with n vertices, i.e., we have found a universal traversal sequence σ'. $\square$

Under the assumption that *linear space* requires exponential size circuits, we can actually construct universal traversal sequences in *logspace*. In fact, that assumption allows us to build logspace computable pseudo-random generators for logspace, and hence to derandomize BPL, the class of languages accepted by logspace randomized Turing machines with bounded two-sided error.

Theorem 5.17 *If there is a Boolean function $f \in \mathsf{DSPACE}[n]$ such that $C_f(n) = 2^{\Omega(n)}$, then $\mathsf{BPL} = \mathsf{L}$.*

Along the lines of Babai et al. [BNS92], the pseudo-random generators behind Theorem 5.17 let us conclude:

Corollary 5.18 *If there is a Boolean function $f \in \mathsf{DSPACE}[n]$ such that $C_f(n) = 2^{\Omega(n)}$, then we can construct universal traversal sequences in logspace.*

Remark: We can actually relax the hypothesis in both Theorem 5.17 and Corollary 5.18 to the existence of a Boolean function in linear space that requires *branching programs* of size $2^{\Omega(n)}$:

If there is a Boolean function $f \in \mathsf{DSPACE}[n]$ that requires branching programs of size $2^{\Omega(n)}$ then $\mathsf{BPL} = \mathsf{L}$ and we can construct universal traversal sequences in logspace.

This stronger statement forms a more natural space-bounded analog of the Impagliazzo-Wigderson [IW97] result than Theorem 5.17. The proof of the statement goes along the lines of the proof of Theorem 3.4. However, it needs a more space efficient transformation of worst-case hardness into average-case hardness than the one by Sudan et al. [STV00] which we used in the proof of Theorem 3.4. The original construction by Impagliazzo and Wigderson [IW97], building on previous work by Babai et al. [BFNW93] and by Impagliazzo [Imp95], would do. Instead of going through the details of the latter constructions, we provide the details for the weaker statement given in Theorem 5.17, for which the Sudan et al. construction used in our proof of Theorem 3.4 works fine.

Proof of Theorem 5.17:
In order to establish Theorem 5.17 it suffices to show the following: If the function f in Theorem 3.4 is in linear space, and $\ell(n) = \Theta(\log n)$, then we can make the pseudo-random generator G provided by Theorem 3.4 computable in linear space. This is because the seed length $s(n)$ of G in Theorem 3.4 is logarithmic for logarithmic $\ell(n)$.

The proof of Theorem 3.4 consists of two steps:

- Converting the worst-case hard function f into an average-case hard function g using Theorem 3.3.
- Applying Theorem 3.2 to the average-case hard function g to obtain the pseudo-random generator G.

First we argue that if the function f in Theorem 3.3 is in linear space then so is the function g. This is because the code $\mathcal{C}_{N,\epsilon}$ in Theorem 3.15 is actually computable in space $O(\log N + \log 1/\epsilon)$, as follows from a straightforward analysis of the paper by Sudan et al. [STV00].

We next apply Theorem 3.2 to the function g in order to obtain the pseudo-random generator G. Computing the ith bit of $G(\sigma)$ for $\sigma \in \{0,1\}^{s(n)}$ amounts to computing the set S_i of a $(\log n, \ell)$ design $\mathcal{S} = (S_1, S_2, \ldots, S_n)$ over $\{1, 2, \ldots, s(n)\}$, and evaluating g on input $\sigma|_{S_i}$. The latter can be done in space $O(s(n))$ since g is in linear space.

The greedy design construction of Lemma 3.12 requires too much space, namely $\Theta(n)$. However, the next lemma describes a different approach which only needs $O(s(n)) = O(\log n)$ space. The construction goes along the lines of Impagliazzo and Wigderson [IW97]. Allender informed us that Wigderson showed him the same construction [ARZ99]. For any $\ell(n) = \Theta(\log n)$, it yields a $(\log n, \ell(n))$ design of size n^δ over $\{1, 2, \ldots, s(n)\}$ for some positive δ and $s(n) = O(\log n)$. Note that this design is smaller than the one obtained through the greedy construction, which yields $\delta = 1$. However, designs of the former type are good enough for Theorem 3.13 (with appropriately modified parameters) and Theorem 3.2 to carry through. So, the next lemma finishes the proof of Theorem 5.17.

Lemma 5.19 *For any positive constant c, there are positive constants α and γ such that we can generate an $(\alpha s, c\alpha s)$ design of size $2^{\gamma s}$ over $\{1, 2, \ldots, s\}$ in space $O(s)$.*

Proof of Lemma 5.19:
We will show that the following process has a positive probability of generating an (m, ℓ) design of size $k \doteq 2^{\gamma s}$ over $\Omega = \{1, 2, \ldots, s\}$ for sufficiently small positive constants α and γ, where $m \doteq \alpha s$ and $\ell \doteq \beta s$ with $\beta \doteq c\alpha$: Pick k subsets of Ω of size ℓ in a pairwise independent way such that each of the $\binom{s}{\ell}$ subsets has about the same probability of being selected.

More precisely, we will do the following. We choose an integer $a \geqslant 0$ such that $2^a \leqslant k^3 \cdot \binom{s}{\ell} < 2^{a+1}$, pick k numbers $r_i \in \{0, 1, \ldots, 2^a - 1\}$ at random in a pairwise independent way, and set S_i to be the $[(r_i \bmod \binom{s}{\ell}) + 1]$-st subset of Ω of size ℓ (say using the lexicographical order). Known constructions of such sample spaces [CG89] only need $O(a)$ random bits and can generate the samples from the random bits in space $O(a)$. Moreover, checking whether a given sample $S_1, \ldots, S_k$ forms an (m, ℓ) design can be done within the same space bounds. Therefore, we can cycle through all possible random bit sequences, check the corresponding candidate design and output the first valid one. This process runs in space $O(a) = O(s)$, and succeeds provided the probability of picking a valid design this way is positive. The remainder of the proof will argue the latter.

Claim 5.20 *Let S, S' be subsets of Ω of size ℓ chosen independently and uniformly at random. Then for some constants d_β and $d' > 0$,*

$$\Pr\left[\,|S \cap S'| \geqslant 2\beta^2 s\,\right] \leqslant d_\beta \cdot s \cdot \exp(-d'\beta^2 s).$$

Proof of Claim 5.20:
First consider a subset S of Ω obtained by doing the following independently for every $i \in \Omega$: Put i in S with probability β. Similarly, and independently, construct S'. Then, by Chernoff's bounds [MR97, Theorem 4.1],

$$\Pr\left[\,|S \cap S'| \geqslant 2\beta^2 s\,\right] \leqslant \exp(-d'\beta^2 s)$$

for some constant $d' > 0$. Stirling's formula yields that

$$\begin{aligned}\Pr\left[\,|S| = \beta s\,\right] &= \binom{s}{\beta s} \cdot \beta^{\beta s}(1-\beta)^{s-\beta s} \\ &\sim \frac{1}{\sqrt{2\pi\beta(1-\beta)}} \cdot \frac{1}{\sqrt{s}}.\end{aligned}$$

So,

$$\begin{aligned}Pr\left[\,|S \cap S'| \geqslant 2\beta^2 s \,\middle|\, |S| = |S'| = \beta s\right] &\leqslant \Pr\left[\,|S \cap S'| \geqslant 2\beta^2 s\,\right] \cdot \Pr\left[\,|S| = |S'| = \beta s\,\right] \\ &\leqslant d_\beta \cdot s \cdot \exp(-d'\beta^2 s)\end{aligned}$$

for some constant d_β depending on β. Since the above distribution of S and S' conditioned on $|S| = |S'| = \beta s$ coincides with the uniform distribution of the statement of the claim, this finishes the proof of the claim. (Claim 5.20) □

There is a constant $d < 2k^3$ such that for any $1 \leqslant i \leqslant k$ and any $T \subseteq \Omega$ with $|T| = \ell$,

$$\frac{d}{2^a} \leqslant \Pr[S_i = T] \leqslant \frac{d+1}{2^a}.$$

Because of the pairwise independence, it follows that for any $1 \leqslant i < j \leqslant k$, the distribution of (S_i, S_j) differs from uniform by at most

$$\begin{aligned}\binom{s}{\ell}^2 \cdot \left[\left(\frac{d+1}{2^a}\right)^2 - \left(\frac{d}{2^a}\right)^2\right] &= (2d+1)\left(\frac{\binom{s}{\ell}}{2^a}\right)^2 \\ &< (2c+1)\left(\frac{2}{k^3}\right)^2 \\ &< \frac{4(4k^3+1)}{k^6} \\ &< \frac{17}{k^3}\end{aligned}$$

in L_1-norm. Therefore, by Claim 5.20,

$$\Pr\left[\,|S_i \cap S_j| \geqslant 2\beta^2 s\,\right] \leqslant \frac{17}{k^3} + d_\beta \cdot s \cdot \exp(-d'\beta^2 s).$$

Recall that $m \doteq \alpha s$, $\ell \doteq \beta s$, and $\beta \doteq c\alpha$. Hence, for $\alpha \leqslant 1/(2c^2)$,

$$\Pr[S_1, S_2, \ldots, S_k \text{ is not an } (m,\ell) \text{ design}] \leqslant \binom{k}{2} \cdot \left(\frac{17}{k^3} + d_\beta \cdot s \cdot \exp(-d'\beta^2 s)\right). \tag{17}$$

The right-hand side of (17) approaches 0 for $k = 2^{\gamma s}$ provided $0 < \gamma < \frac{d'\beta^2}{\ln 4} = \frac{d' c^2 \alpha^2}{\ln 4}$. This finishes the proof of the lemma. (Lemma 5.19) □

6 Recent Progress and Further Research

In this paper, we have demonstrated the power of relativization in the area of derandomization. We gave several striking examples, most notably Arthur-Merlin games. New applications have been discovered recently, and we are convinced that even more will follow.

Critical to our argument is the idea that the analyses of the Nisan-Wigderson and other generators work for a large class of predicates and statistical tests – whenever the generator fails a statistical test, the computation of the predicate on which the generator is based reduces to the computation of the test. This idea lies at the heart of the breakthrough extractor constructions of Trevisan [Tre99]. In fact, following Trevisan [Tre99], Miltersen[Mil98] explicitly describes the role of relativization there. He argues that every relativizable construction of a pseudo-random generator out of a hard Boolean function yields an extractor. We showed that the Impagliazzo-Wigderson construction relativizes. Applying Miltersen's argument to the latter yields Trevisan's extractors.

Another application of relativization along the lines of our work is the following. Nisan and Wigderson [NW94] observed that if E has circuits of size $2^{o(n)}$ then for every function $t(n) = 2^{\Omega(n)}$ and every positive constant ϵ

$$\mathsf{DTIME}[t] \subseteq \mathsf{DSPACE}[t^{\epsilon}]. \tag{18}$$

They concluded from their main results that every language in BPP can be decided in deterministic polynomial time for infinitely many input lengths unless (18) holds. Lu [Lu00] noticed that the inclusion (18) even follows if E has B-oracle circuits of size $2^{o(n)}$ for some oracle B that is decidable in linear space. Therefore, by setting $B = \mathrm{SAT}$, the weak version of Theorem 3.10 implies that every language in AM has polynomial-size proofs for infinitely many input lengths unless (18) holds [Lu00].

A different line of further research consists of relaxing the hardness condition for derandomizing Arthur-Merlin games. In particular, we would like to replace the SAT-oracle circuit model by the weaker nondeterministic circuit model in Theorem 3.8. Miltersen and Vinodchandran recently managed to do this for the special case of Theorem 3.10. They showed that if $\mathsf{NE} \cap \mathsf{coNE}$ requires nondeterministic circuits of size $2^{\Omega(n)}$ then AM collapses to NP. They use a different technique, suited for one-sided error classes, and exploit the fact that all languages in AM possess a one-sided error bounded round Arthur-Merlin game. More specifically, they prove that if there exists a Boolean function $f \in \mathsf{E}^{\mathcal{A}}$ that requires nondeterministic B-oracle circuits of size $2^{\Omega(n)}$ then there exists a hitting set generator computable in $\mathsf{E}^{\mathcal{A}}$ with logarithmic seed length that works for conondeterministic B-oracle circuits. Instantiating this statement with $\mathcal{A} = \mathsf{NP} \cap \mathsf{coNP}$ and $B = \varnothing$ yields their main result. In a similar way, the SAT-oracle circuits in Section 5.3 on rigid matrices can be replaced by nondeterministic circuits [MV99], and the TQBF_2-oracle circuits in Section 5.2 on exact learning can be replaced by nondeterministic SAT-oracle circuits.

Due to its one-sided character, the technique of Miltersen and Vinodchandran does not apply to our results in Section 5.1 on derandomizing the Valiant-Vazirani hashing procedure. It also only works at the "high-end" of the derandomization spectrum, i.e., under strong hardness assumptions as in the last lines of Tables 1 and 2. The weakest hardness assumption Miltersen and Vinodchandran can handle is the existence of a Boolean function in $\mathsf{NEXP} \cap \mathsf{coNEXP}$ that requires nondeterministic circuits of size $2^{n^{\alpha}}$ for some constant $\alpha > \frac{1}{2}$ (then they obtain quasipolynomial size proofs for any language in AM, as in Theorem 3.9). In particular, it remains open whether the existence of a Boolean function $f \in \mathsf{NEXP} \cap \mathsf{coNEXP}$ that requires superpolynomial size nondeterministic circuits implies that graph nonisomorphism has subexponential size proofs. We present this as an open problem for further research.

Acknowledgments

We want to express our sincere gratitude to Oded Goldreich, Madhu Sudan, Luca Trevisan, and Salil Vadhan for their encouragement and collaboration. We would also like to thank Rudi Seitz for his contribution to Section 5.4; Theorem 5.16 was proved jointly with him.

The first author wishes to thank Dan Spielman and Salil Vadhan for their support on all aspects of this paper. The first author also wishes to thank Henry Cohn, Steven Rudich, and Avi Wigderson for important conversations.

The second author wishes to thank his adviser Lance Fortnow for the guidance, suggestions, and support he received, and for the statement of Corollary 5.10 in particular. He also wants to thank the following people for stimulating and helpful discussions about issues related to this paper: Eric Allender, Laci Babai, Behfar Bastani, Harry Buhrman, Valentine Kabanets, Peter Bro Miltersen, Mike Saks, Madhu Sudan, Avi Wigderson, and Shiyu Zhou. The second author is grateful to Eric Allender, Peter Bro Miltersen, Valentine Kabanets, and Salil Vadhan for the comments he received on an earlier version of the paper [vM98].

The authors would also like to thank the journal referees for their helpful suggestions and comments.

References

[Adl78] L. Adleman. Two theorems on random polynomial time. In *Proceedings of the 19th IEEE Symposium on Foundations of Computer Science*, pages 75–83. IEEE, 1978.

[AK97] V. Arvind and J. Köbler. On pseudorandomness and resource-bounded measure. In *Proceedings 17th Conference on the Foundations of Software Technology and Theoretical Computer Science*, volume 1346 of *Lecture Notes in Computer Science*, pages 235–249. Springer-Verlag, 1997.

[AKL+79] R. Aleliunas, R. Karp, R. Lipton, L. Lovász, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science*, pages 218–223. IEEE, 1979.

[Ang88] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

[ARZ99] E. Allender, K. Reinhardt, and S. Zhou. Isolation, matching, and counting: Uniform and nonuniform upper bounds. *Journal of Computer and System Sciences*, 59:164–181, 1999.

[Bab85] L. Babai. Trading group theory for randomness. In *Proceedings of the 17th ACM Symposium on the Theory of Computing*, pages 421–429. ACM, 1985.

[BCG+96] N. Bshouty, R. Cleve, R. Gavaldà, S. Kannan, and C. Tamon. Oracles and queries that are sufficient for exact learning. *Journal of Computer and System Sciences*, 52(3):421–433, 1996.

[BDG95] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*, volume 11 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1995.

[BFNW93] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993.

[BKL83] L. Babai, W. Kantor, and E. Luks. Computational complexity and the classification of finite simple groups. In *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, pages 162–171. IEEE, 1983.

[BL83] L. Babai and E. Luks. Canonical labeling of graphs. In *Proceedings of the 15th ACM Symposium on the Theory of Computing*, pages 171–183. ACM, 1983.

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13:850–864, 1984.

[BM88] L. Babai and S. Moran. Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36:254–276, 1988.

[BNS92] L. Babai, N. Nisan, and M. Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *Journal of Computer and System Sciences*, 45:204–232, 1992.

[CG89] B. Chor and O. Goldreich. On the power of two-point based sampling. *Journal of Complexity*, 5:96–106, 1989.

[Coo71] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd ACM Symposium on the Theory of Computing*, pages 151–158. ACM, 1971.

[Coo88] S. Cook. Short propositional formulas represent nondeterministic computations. *Information Processing Letters*, 26:269–270, 1988.

[CRT98] A. Clementi, J. Rolim, and L. Trevisan. Recent advances towards proving P = BPP. *Bulletin of the European Association for Theoretical Computer Science*, 64:96–103, February 1998.

[FFK94] S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. *Journal of Computer and System Sciences*, 48(1):116–148, 1994.

[FGM+89] M. Fürer, O. Goldreich, Y. Mansour, M. Sipser, and S. Zachos. On completeness and soundness in interactive proof systems. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 429–442. JAI Press, Greenwich, 1989.

[For97] L. Fortnow. Counting complexity. In L. Hemaspaandra and A. Selman, editors, *Complexity Theory Retrospective II*, pages 81–107. Springer-Verlag, 1997.

[Fri90] J. Friedman. A note on matrix rigidity. Technical Report TR-308-91, Department of Computer Science, Princeton University, 1990.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[GMW91] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38:691–729, 1991.

[GS89] S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof systems. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 73–90. JAI Press, Greenwich, 1989.

[Imp95] R. Impagliazzo. Hard-core distributions for somewhat hard problems. In *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science*, pages 538–545. IEEE, 1995.

[IW97] R. Impagliazzo and A. Wigderson. P=BPP unless E has sub-exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th ACM Symposium on the Theory of Computing*, pages 220–229. ACM, 1997.

[KL82] R. Karp and R. Lipton. Turing machines that take advice. *L'Enseignement Mathématique*, 28(2):191–209, 1982. A preliminary version appeared in STOC 1980.

[KvM99] A. Klivans and D. van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial hierarchy collapses. In *Proceedings of the 31st ACM Symposium on the Theory of Computing*, pages 659–667. ACM, 1999.

[Lit88] N. Littlestone. Learning when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[LP82] H. Lewis and C. Papadimitriou. Symmetric space-bounded computation. *Theoretical Computer Science*, 19(2):161–187, 1982.

[Lu00] C.-J. Lu. Derandomizing Arthur-Merlin games under uniform assumptions. In *Proceedings of the Eleventh International Symposium on Algorithms and Computation*. Springer-Verlag, 2000. To appear.

[Mil98] P. Bro Miltersen. Relativizable pseudorandom generators and extractors. Comment to ECCC Technical Report TR98-055, 1998.

[MR97] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1997.

[MV99] P. Bro Miltersen and N. Vinodchandran. Derandomizing Arthur-Merlin games using hitting sets. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, pages 71–80. IEEE, 1999.

[Nis92] N. Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12:449–461, 1992.

[NT98] N. Nisan and A. Ta-Shma Extracting Randomness: A Survey and New Constructions. *Journal of Computer and System Sciences*, 1998.

[NW94] N. Nisan and A. Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.

[Pap94] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[PV91] P. Pudlák and Z. Vavřín. Computation of rigidity of order n^2/r for one simple matrix. *Comment. Math. Univ. Carolinae*, 32:213–218, 1991.

[PW90] L. Pitt and M. Warmuth. Prediction preserving reducibility. *Journal of Computer and System Sciences*, 41(3):430–467, 1990.

[PZ83] C. Papadimitriou and S. Zachos. Two remarks on the power of counting. In *Proceedings of the 6th GI-Conference on Theoretical Computer Science*, volume 145 of *Lecture Notes in Computer Science*, pages 269–276. Springer-Verlag, 1983.

[Raz] A. Razborov. On rigid matrices. Manuscript in Russian.

[Rud97] S. Rudich. Super-bits, demi-bits, and $\tilde{\mathrm{NP}}$/qpoly-natural proofs. In *Proceedings of the 1st International Symposium on Randomization and Approximation Techniques in Computer Science*, volume 1269 of *Lecture Notes in Computer Science*, pages 85–93. Springer-Verlag, 1997.

[STV00] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. Technical Report TR-98-074, Electronic Colloquium on Computational Complexity, 2000. Revision 2.

[TO92] S. Toda and M. Ogiwara. Counting classes are at least as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 21(2):316–328, 1992.

[Tod91] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[Tre99] L. Trevisan. Construction of extractors using pseudo-random generators. In *Proceedings of the 31st ACM Symposium on the Theory of Computing*, pages 141–148. ACM, 1999.

[Val77] L. Valiant. Graph-theoretic arguments in low-level complexity. In *Proceedings of the 6th Symposium on Mathematical Foundations of Computer Science*, volume 53 of *Lecture Notes in Computer Science*, pages 162–176. Springer-Verlag, 1977.

[vM98] D. van Melkebeek. Derandomizing Arthur-Merlin games. Technical Report TR-98-08, Department of Computer Science, The University of Chicago, 1998.

[VV86] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.

[Yao82] A. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, pages 80–91. IEEE, 1982.