Paradigms for Unconditional Pseudorandom Generators

Other titles in Foundations and Trends® in Theoretical Computer Science

Approximate Degree in Classical and Quantum Computing
Mark Bun and Justin Thaler
ISBN: 978-1-63828-140-5

Multi-Valued Reasoning about Reactive Systems
Orna Kupferman
ISBN: 978-1-63828-138-2

Quantified Derandomization: How to Find Water in the Ocean
Roei Tell
ISBN: 978-1-63828-092-7

Complexity Theory, Game Theory, and Economics: The Barbados Lectures
Tim Roughgarden
ISBN: 978-1-68083-654-7

Semialgebraic Proofs and Efficient Algorithm Design
Noah Fleming, Pravesh Kothari and Toniann Pitassi
ISBN: 978-1-68083-636-3

Paradigms for Unconditional Pseudorandom Generators

Pooya Hatami
The Ohio State University
pooyahat@gmail.com

William Hoza
The University of Chicago
williamhoza@uchicago.edu

now
the essence of knowledge
Boston — Delft

Foundations and Trends® in Theoretical Computer Science

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

P. Hatami and W. Hoza. *Paradigms for Unconditional Pseudorandom Generators.* Foundations and Trends® in Theoretical Computer Science, vol. 16, no. 1-2, pp. 1–210, 2024.

ISBN: 978-1-63828-335-5

Foundations and Trends® in Theoretical Computer Science
Volume 16, Issue 1-2, 2024

Editorial Board

Editorial Scope

Topics

Foundations and Trends® in Theoretical Computer Science publishes survey and tutorial articles in the following topics:

- Algorithmic game theory
- Computational algebra
- Computational aspects of combinatorics and graph theory
- Computational aspects of communication
- Computational biology
- Computational complexity
- Computational geometry
- Computational learning
- Computational Models and Complexity
- Computational Number Theory
- Cryptography and information security
- Data structures
- Database theory
- Design and analysis of algorithms
- Distributed computing
- Information retrieval
- Operations Research
- Parallel algorithms
- Quantum Computation
- Randomness in Computation

Information for Librarians

Foundations and Trends® in Theoretical Computer Science, 2024, Volume 16, 4 issues. ISSN paper version 1551-305X. ISSN online version 1551-3068. Also available as a combined paper and online subscription.

Contents

Paradigms for Unconditional Pseudorandom Generators

Pooya Hatami[1] and William Hoza[2]

[1] *The Ohio State University, USA; pooyahat@gmail.com*
[2] *The University of Chicago, USA; williamhoza@uchicago.edu*

ABSTRACT

This is a survey of unconditional *pseudorandom generators* (PRGs). A PRG uses a short, truly random seed to generate a long, "pseudorandom" sequence of bits. To be more specific, for each restricted model of computation (e.g., bounded-depth circuits or read-once branching programs), we would like to design a PRG that "fools" the model, meaning that every function computable in the model behaves approximately the same when we plug in pseudorandom bits from the PRG as it does when we plug in truly random bits. In this survey, we discuss four major paradigms for designing PRGs:

- We present several PRGs based on k-wise uniform generators, small-bias generators, and simple combinations thereof, including proofs of Viola's theorem on fooling low-degree polynomials [242] and Braverman's theorem on fooling $\mathbf{AC}^0$ circuits [36].
- We present several PRGs based on "recycling" random bits to take advantage of communication bottlenecks, such as the Impagliazzo-Nisan-Wigderson generator [131].

Pooya Hatami and William Hoza (2024), "Paradigms for Unconditional Pseudorandom Generators", Foundations and Trends® in Theoretical Computer Science: Vol. 16, No. 1-2, pp 1–210. DOI: 10.1561/0400000109.

- We present connections between PRGs and computational hardness, including the Nisan-Wigderson framework for converting a hard Boolean function into a PRG [183].
- We present PRG frameworks based on random restrictions, including the "polarizing random walks" framework [49].

We explain how to use these paradigms to construct PRGs that work *unconditionally*, with no unproven complexity-theoretic assumptions. The PRG constructions use ingredients such as finite field arithmetic, expander graphs, and randomness extractors. The analyses use techniques such as Fourier analysis, sandwiching approximators, and simplification-under-restrictions lemmas.

1

Introduction

To make random choices, it would be useful to have an unlimited supply of "truly random" bits: unbiased and independent coin flips. What can we do if we only have a few truly random bits? A *pseudorandom generator* (PRG) uses a small amount of true randomness, called the "seed," to generate a long sequence that appears to be completely random (even though it isn't). PRGs are ubiquitous in computing theory and practice. The basic motivation is that we think of randomness as a scarce computational resource, akin to time or space, so whenever we get our hands on some random bits, we want to stretch them as far as possible.

To model PRGs mathematically, we consider some "observer," modeled as a function f. Let U_n denote the uniform distribution over $\{0,1\}^n$. We would like to "fool" f in the following sense.

Definition 1.1 (Fooling)**.** Suppose $f\colon \{0,1\}^n \to \{0,1\}$ is a function, X is a probability distribution over $\{0,1\}^n$, and $\varepsilon > 0$. We say that X *fools* f *with error* ε, or ε*-fools* f, if

$$|\Pr[f(X) = 1] - \Pr[f(U_n) = 1]| \leq \varepsilon.$$

More generally, we can consider a real-valued function $f\colon \{0,1\}^n \to \mathbb{R}$. In this case, we say that X fools f with error ε if

$$|\,\mathbb{E}[f(X)] - \mathbb{E}[f(U_n)]| \leq \varepsilon.$$

If $\varepsilon = 0$, we say that X *perfectly* fools f.

Remark 1.1. As a shorthand, we often identify the function f with the random variable $f(U_n)$. For example, instead of $\mathbb{E}[f(U_n)]$, we simply write $\mathbb{E}[f]$.

Definition 1.1 says that although X might not be uniform, X and U_n are nevertheless *indistinguishable*, at least from f's perspective. Conversely, if X does *not* ε-fool f, we refer to f as a "distinguisher" for X. A PRG's job is to use a few truly random bits to sample a distribution that fools f.

Definition 1.2 (PRGs)**.** Suppose $f\colon \{0,1\}^n \to \mathbb{R}$ and $G\colon \{0,1\}^s \to \{0,1\}^n$ are functions and $\varepsilon > 0$. We say that G is an *ε-PRG for f* if $G(U_s)$ fools f with error ε. In this case, we also say that G fools f with error ε (see Figure 1.1.)

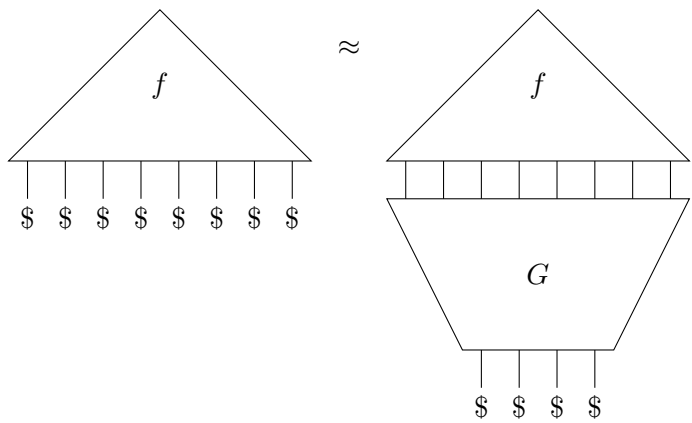

Figure 1.1: A PRG (G) uses a few truly random bits (depicted here using \$ symbols) to sample a pseudorandom string that is indistinguishable from a truly random string, from the perspective of the observer (f).

The parameter s is called the *seed length* of the PRG; we would like s to be as small as possible. Throughout this text, the parameter "n" will always denote the number of pseudorandom bits we are generating.

1.1 Whom Shall We Fool? Three Approaches to PRGs

An unavoidable fact of life is that for any nontrivial PRG, there exists a function that is not fooled by the PRG.

Claim 1.1 (Impossibility of fooling all functions)**.** Let $G\colon \{0,1\}^s \to \{0,1\}^n$ where $s < n$. There exists some $f\colon \{0,1\}^n \to \{0,1\}$ such that G does not 0.49-fool f.

Proof. Let f be the indicator function for the image of G. Then $\mathbb{E}[f(G(U_s))] = 1$, whereas $\mathbb{E}[f] \leq 1/2$ because $s < n$. □

In light of Claim 1.1, the best we can hope for is generating bits that fool some large sets of observers but not all of them. After all, as Avi Wigderson says, *randomness is in the eye of the beholder* [248].

Definition 1.3 (PRG for a class of functions)**.** Let $n \in \mathbb{N}$, let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \mathbb{R}$, let $G\colon \{0,1\}^s \to \{0,1\}^n$ be a function, and let $\varepsilon > 0$. We say that G is an *ε-PRG for $\mathcal{F}$* if G fools every $f \in \mathcal{F}$ with error ε.

Which observers shall we fool? The study of PRGs can be crudely divided into three approaches based on three possible answers:

1. Everyday non-adversarial applications.
2. All efficient observers.
3. Restricted models of computation.

We discuss these three approaches in Sections 1.1.1 to 1.1.3.

1.1.1 PRGs for everyday non-adversarial applications

In practice, when programmers want randomness, they invoke some type of `random()` method provided by the computing environment. Under the hood, these `random()` methods typically involve several components, each of which might be quite sophisticated. When practitioners speak of "pseudorandom number generators" or "random number generators," they are usually referring to the entire randomness system as a whole,

including whatever techniques are used to produce an initial seed. For example, the system might derive a seed from the current time of day, even though such a seed is rather predictable. As another example, the system might use hardware random number generators based on thermal noise measurements.

In this text, we sidestep the important issue of producing a seed, along with many other issues that are important in practice. We focus on the challenge of stretching a truly random seed out to a long pseudorandom string. In our terminology, this is the job of a PRG (see Definition 1.2). A PRG is thus one of multiple components of a practical randomness system. For example, Java's `Math.random()` method currently uses a type of PRG called a *linear congruential generator*. For such a PRG, the seed is a random number $X_0 \in \{0, 1, \ldots, M-1\}$, and the output sequence is $(X_1, X_2, X_3, \ldots)$, where

$$X_{i+1} = a \cdot X_i + b \bmod M$$

for some parameters M, a, b. Meanwhile, Python's `random.random()` method uses an algorithm called the "Mersenne twister" [169], and major web browsers currently use a PRG in the "xorshift+ family" [240] to implement Javascript's `Math.random()` function.

Why these PRGs are unsatisfactory

Practitioners use these randomness systems for both casual applications (e.g., video games) and serious applications (e.g., scientific simulations). However, for a generic randomized algorithm, there is *no firm mathematical guarantee* that the outputs will be reliable when the algorithm is executed using one of these practical randomness systems. The methods that practitioners typically use to run randomized algorithms must be considered *heuristics.*

To be clear, a lot of work goes into designing high-quality practical randomness systems. Designers strive to ensure that these systems can be safely used in any application that "comes up naturally" in practice. The system is only deemed acceptable for everyday use when it passes a great number of creative statistical tests, such as those in the TestU01 family [147].

These statistical tests are valuable, but there is a wide gap between the statistical tests and a typical randomized algorithm. The designers behind practical systems such as Java's `Math.random()` method wisely do not claim that they work in *adversarial* scenarios, so these systems are considered unsuitable for cryptography. This is true even if we focus solely on the PRG component of these systems. Furthermore, sometimes programs "accidentally" distinguish pseudorandom numbers from truly random numbers. There are quite a few documented cases in which PRGs have been shown to cause inaccurate scientific simulations [68], [69], [88], [89], [107], [139], [174], [187]! One must imagine that other cases have gone unnoticed.

To a theoretician, this state of affairs is deeply unsatisfactory. Yes, modern practical PRGs seem to almost always work well in practice, but we don't have a mathematically rigorous explanation for *why* these systems work. It's not even clear what precisely the goal is. (Mathematically, how can we make a distinction between "adversarially-designed" programs and "naturally-occurring" programs?) By theoreticians' standards, the success of practical PRGs is largely a mystery.

1.1.2 PRGs for all efficient observers

One of the great ideas in the theory of computing is the concept of a PRG that fools *all computationally efficient* observers. Given such a PRG and a truly random seed, we would be able to execute any randomized algorithm that is actually worth executing. (After all, there's no point running a program if one won't even survive long enough to see the output!) Such a PRG could also be used in cryptographic settings, because we can safely assume that eavesdroppers and hackers only have so much computational power.[1]

For example, the Blum-Blum-Shub (BBS) generator [27] uses a short seed to randomly select a suitable modulus M and a number

[1]There is a subtle distinction here. In the context of randomized algorithms, it's okay if the PRG itself uses a little more time than the algorithms that we are trying to fool. On the other hand, in the context of cryptography, we want an efficiently-computable PRG that fools all efficient adversaries, including those that use polynomially more time than the PRG uses.

$X_0 \in \{1, 2, \ldots, M-1\}$, and then it outputs the sequence $(X_1 \bmod 2, X_2 \bmod 2, X_3 \bmod 2, \ldots)$ where

$$X_{i+1} = X_i^2 \bmod M.$$

This PRG is reminiscent of linear congruential generators, but the similarity is only superficial. It is believed that the BBS generator fools polynomial-time algorithms.

Why these PRGs are also (currently) unsatisfactory

Fooling all efficient observers is a well-defined and well-motivated *goal*. Unfortunately, nobody knows how to prove that some efficiently-computable PRG actually *has* this marvelous property.

To be clear, there is a substantial body of "evidence" indicating that such PRGs exist. For example, Blum *et al.* [27] showed that their generator fools all polynomial-time observers, under the plausible-but-unproven assumption that there is no good algorithm for the "quadratic residuosity problem". There are many other examples of PRGs that fool all polynomial-time observers under reasonable cryptographic or complexity-theoretic assumptions [28], [83], [119], [134], [144], [183], [236], [249].[2] For practical cryptography, software developers tend to use PRGs that are not even supported by rigorous *conditional* proofs of correctness, but rather are supported by heuristic and intuitive arguments.

There is a genuine possibility that these PRGs are not secure. In one infamous incident, the U.S. National Institute of Standards and Technology (NIST) recommended using a PRG called "`Dual_EC_DRBG`." The PRG was designed by the U.S. National Security Agency (NSA), and allegedly, they *intentionally designed it to be insecure* for surveillance purposes [189].

Once again, to a theoretician, this state of affairs is not satisfactory. There is genuine room for doubt about *whether* known PRGs work, and perhaps more importantly, even if they do work, we don't have a good

[2] Note that some of these PRGs use somewhat more time than the observers they fool, and hence are suitable for simulating randomized algorithms but not for cryptography (cf. Footnote 1).

explanation for *why* they work. Conditional proofs can be considered partial explanations at best. The problem of designing PRGs that unconditionally fool all efficient observers is very challenging, with connections to deep topics such as the famous **P** vs. **NP** problem (see Section 4.1).

1.1.3 PRGs for restricted models of computation

The main topic of this text is a third approach to PRGs. In this third approach, we identify an interesting and well-defined *restricted model of computation*. Then we design PRGs that fool the chosen model of computation (unconditionally – with no unproven assumptions) and try to optimize the seed length of the PRG.

A toy example might clarify the idea. Let us design a PRG

$$G\colon \{0,1\}^2 \to \{0,1\}^3$$

that fools every observer f that only looks at two of the three output bits. This problem is not completely trivial, because we don't know which two bits f will observe. Nevertheless, the problem can be solved by defining

$$G(u_1, u_2) = (u_1, u_2, u_1 \oplus u_2),$$

where $\oplus$ denotes the XOR operation. When u_1 and u_2 are chosen uniformly at random, the three output bits are correlated, but any two of the bits are independent and uniform random.

Unconditional PRGs can be constructed for much richer and more interesting restricted models of computation. We are especially interested in fooling models of computation that have a "complexity theory" flavor, i.e., we want the output of the PRG to appear random to any observer that is "sufficiently efficient" in some sense. Arguably, the two most important models in this field are *constant-depth circuits* ($\mathbf{AC}^0$, see Definition 2.13) and *read-once branching programs* (ROBPs, see Definition 1.5).

The value of these PRGs

Could PRGs for restricted models ever be directly used in practical applications? Potentially. PRGs for restricted models can be used to

simulate randomized algorithms without significantly distorting their behavior, provided that the algorithms in question are "sufficiently efficient" in the appropriate sense. (See Section 1.5 for more details.)

Admittedly, it's a bit unrealistic to imagine the PRGs studied in the theoretical literature being implemented on actual computers, because it is hard to compete with the practical PRGs discussed in Section 1.1.1. Instead, the study of PRGs for restricted models has a much grander and broader purpose: these PRGs help to uncover the mysteries of the theory of computing, and hence are invaluable from a scientific perspective.

We briefly elaborate on some of the applications of PRGs within the theory of computing in Section 1.5. Apart from any application, we hope to convince the reader that PRGs for restricted models are interesting in their own right.

1.2 Overview of this Text

In this work, we survey some of the most important frameworks and techniques for constructing unconditional PRGs for restricted models of computation. We focus on four major PRG paradigms:

- In Section 2, we present k-wise uniform generators, small-bias generators, and simple combinations thereof.
- In Section 3, we present PRGs that "recycle" randomness to take advantage of communication bottlenecks, such as the Impagliazzo-Nisan-Wigderson generator [131].
- In Section 4, we present connections between PRGs and computational hardness, including the Nisan-Wigderson framework for converting a hard Boolean function into a PRG [183].
- In Section 5, we present methods for constructing PRGs based on (pseudo)random restrictions, including the relatively recent "polarizing random walks" framework [49].

Along the way, as needed, we introduce the computational models that we fool (decision trees, circuits, branching programs, etc.) and tech-

niques for analyzing PRGs (Fourier analysis, sandwiching approximators, simplification-under-restriction lemmas, etc.)

The literature on unconditional PRGs is vast, and this survey is far from exhaustive. (For example, we do not discuss the important line of work on fooling *linear threshold functions* [78], [104], [173], [195].) Instead, we hope that this work serves as a suitable *introduction* to the field of unconditional PRGs, preparing the reader to study new and old papers on PRGs and make their own contributions.

The results that we cover include both classic and recent works. Besides covering the most important *principles* of PRG design and analysis, we also made sure to include expositions of many of the most important *examples* of unconditional PRGs, such as Viola's [242] PRG for low-degree polynomials, Braverman's [36] theorem that limited independence fools $\mathbf{AC}^0$, and Forbes and Kelley's [91] relatively recent PRG for arbitrary-order ROBPs.

This text is primarily expository. However, we couldn't help but include a few novel theorems and proofs. For example, we present a new proof of Braverman's theorem (Section 2.6), and we present a new improvement to the polarizing random walks framework in the low-error regime (Section 5.1.4). We also highlight plenty of important open problems regarding PRGs for restricted models of computation.

Many wonderful prior expository works [15], [97], [165], [175], [185], [238] and lecture notes [45]–[47], [215], [216], [218], [219], [229], [233], [244], [253] include some coverage of unconditional PRGs. However, none of them has quite the same focus as our work, so we feel that our work fills a gap.

In the rest of this section, we discuss some additional basic issues related to the concept of a PRG, paving the way for the PRG constructions in subsequent sections.

1.3 The Generic Probabilistic Existence Proof

For many classes $\mathcal{F}$, including classes defined by standard nonuniform computational models (such as decision trees, circuits, branching programs, etc.), there is a totally generic argument showing that there exist PRGs that fool $\mathcal{F}$ with a small seed length.

Proposition 1.1 (Nonexplicit PRGs)**.** Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$. For every $\varepsilon > 0$, there exists an ε-PRG for $\mathcal{F}$ with seed length $\log\log|\mathcal{F}| + 2\log(1/\varepsilon) + O(1)$.

Proof. Pick a function $G\colon \{0,1\}^s \to \{0,1\}^n$ uniformly at random. Consider any arbitrary $f \in \mathcal{F}$. For each seed y, the value $f(G(y))$ is a random bit satisfying

$$\mathop{\mathbb{E}}_{G}[f(G(y))] = \mathop{\mathbb{E}}_{U_n}[f(U_n)].$$

Furthermore, as y ranges over all 2^s possible seeds, these random variables $f(G(y))$ are independent. Therefore, by Hoeffding's inequality,

$$\Pr_{G}\left[\left|\mathbb{E}[f] - 2^{-s}\sum_{y\in\{0,1\}^s} f(G(y))\right| > \varepsilon\right] \le 2e^{-2\varepsilon^2 2^s}.$$

By the union bound, the probability that G fails to ε-fool $\mathcal{F}$ is bounded by $2|\mathcal{F}|e^{-2\varepsilon^2 2^s}$. For $s = \log\log|\mathcal{F}| + 2\log(1/\varepsilon) + O(1)$, this probability is less than 1, i.e., there exists a G that does ε-fool $\mathcal{F}$. □

In a typical case – e.g., if $\mathcal{F}$ is the set of all circuits of size at most n – each function $f \in \mathcal{F}$ can be described using $\mathrm{poly}(n)$ bits, i.e., $|\mathcal{F}| \le 2^{\mathrm{poly}(n)}$. In this case, the PRG guaranteed by Proposition 1.1 has seed length $O(\log(n/\varepsilon))$.

1.4 Explicitness

Proposition 1.1 has a major weakness: it does not guarantee that the PRG is *efficiently computable*. The proof of Proposition 1.1 is in some sense "nonconstructive." Ideally, we want an *algorithm* for sampling from a pseudorandom distribution, and we want the algorithm to be reasonably efficient with respect to randomness *and* more conventional complexity measures simultaneously.

Definition 1.4 (Explicitness)**.** A PRG $G\colon \{0,1\}^s \to \{0,1\}^n$ is *explicit* if it can be computed in time $\mathrm{poly}(n)$.

One could consider alternative standards of explicitness. We could require that each individual output bit can be computed in time $\mathrm{polylog}\, n$,

or that the PRG runs in space $O(\log n)$, or that each bit can be computed in $\mathbf{AC}^0$, or any number of other conditions. The truth is, there is no "one true definition" of explicitness. The appropriate definition depends on what one hopes to gain from the PRG; see Section 1.5.

In this text, we will stick with Definition 1.4 for concreteness, but when we present PRG constructions, we will generally not bother carefully verifying the runtime bound. Instead, we will focus on making the construction clear to the reader.

1.4.1 Families of PRGs

Definition 1.4 refers to the time complexity of a PRG. To meaningfully speak of time complexity, we technically ought to be considering a whole *family* of PRGs. The convention in this line of work is to keep the family implicit. For example, a theorem might say something like the following.

> For all $n, m \in \mathbb{N}$ and all $\varepsilon > 0$, there exists an explicit ε-PRG for size-m decision trees on n input bits with seed length $O(\log(m/\varepsilon) + \log\log n)$.

(See Section 2.3.3.) Translating into more precise language, the same theorem can be restated as follows.

> There exists a randomized algorithm $\mathcal{G}$ satisfying the following.
>
> 1. Given input parameters n, m, ε, the algorithm $\mathcal{G}$ outputs a string $\mathcal{G}(n, m, \varepsilon) \in \{0,1\}^n$.
> 2. For all n, m, ε, the output distribution $\mathcal{G}(n, m, \varepsilon)$ fools size-m decision trees with error ε.
> 3. $\mathcal{G}(n, m, \varepsilon)$ uses at most $O(\log(m/\varepsilon) + \log\log n)$ random bits and runs in time $\mathrm{poly}(n)$.

There is something potentially troubling about this "translation" process. The quantifiers got flipped! In the informal theorem statement,

we say "for all n, m, ε, there exists an explicit PRG," but strictly speaking, we mean that there exists a single algorithm $\mathcal{G}$ that works for all n, m, ε simultaneously! Is this "flipped quantifiers" convention wise?

Let us make an analogy with big-O notation. Recall, e.g., the famous planar separator theorem:

> For all $n \in \mathbb{N}$, for every n-vertex planar graph, there exists a set of $O(\sqrt{n})$ vertices such that removing those vertices splits the graph into connected components with at most $2n/3$ vertices each.

If we wanted to be more rigorous, we ought to flip the quantifiers and write something like the following:

> There exists a function $f\colon \mathbb{N} \to \mathbb{N}$ such that $f \in O(\sqrt{n})$ and for all $n \in \mathbb{N}$, for every n-vertex planar graph, there exists a set of $f(n)$ vertices such that removing those vertices splits the graph into connected components with at most $2n/3$ vertices each.

We don't bother with such careful language because it obscures more than it clarifies. The important thing is that the expression "$O(\sqrt{n})$" tells how the number of removed vertices scales with the universally quantified parameter n. Analogously, when we say "there exists an explicit PRG," the word "explicit" tells how the *computational complexity* of the PRG scales with the parameters.

1.4.2 The default conjecture: Explicit PRGs exist

For each "reasonable" class $\mathcal{F}$, the standard conjecture is that there exists an *explicit* PRG with essentially the same seed length as the generic nonexplicit bound (Proposition 1.1). Oftentimes, this conjecture can be supported with evidence in the form of conditional constructions. For example, consider the class $\mathcal{F}$ of all CNF formulas of size at most n. The nonexplicit PRG has seed length $O(\log(n/\varepsilon))$. Under plausible complexity-theoretic assumptions, there is indeed an explicit PRG for

all size-n Boolean circuits (whether CNF formulas or not) with seed length $O(\log(n/\varepsilon))$ [134].

Even without a compelling conditional construction, the "default" conjecture would be that for natural families of functions a probabilistic existence proof can be matched by an explicit construction. The main challenge is to find the explicit construction. Typically, such a PRG would be optimal, i.e., one can unconditionally prove a seed length lower bound matching the nonexplicit bound to within a constant factor.[3] For example, every PRG for size-n CNF formulas (explicit or not) must have seed length at least $\Omega(\log(n/\varepsilon))$.

1.5 Applications of PRGs

PRGs for restricted models have many applications. We will not attempt to exhaustively list these applications, nor even to properly survey them. We will, however, briefly describe some of the most important applications. Hopefully, this brief discussion of applications will serve to motivate the main topic of this text, which is the construction and analysis of PRGs.

1.5.1 Simulating randomized algorithms

One of the most natural applications of PRGs is to simulate a randomized algorithm using only a few truly random bits (the seed of the PRG). Let A be a randomized algorithm that we would like to simulate. In order to simulate A without significantly distorting its behavior, what property should our PRG have?

For simplicity, let us assume that A is a decision algorithm, i.e., it outputs a bit. Let $A(a, x)$ denote the output value of A when the input is a and the random bits are x. For each input a, we can define a function $f_a\colon \{0,1\}^n \to \{0,1\}$, where n is the number of random bits that A uses,[4] by the rule $f_a(x) = A(a, x)$. That is, f_a describes the behavior of A on input a as a function of its random bits. Definition 1.2

[3] For a counterexample, see the work of Hoza *et al.* [127].

[4] For simplicity, we assume that n is determined by a rather than varying based on the random bits. This is a "Monte Carlo" algorithm rather than a "Las Vegas" algorithm.

implies that if $G\colon \{0,1\}^s \to \{0,1\}^n$ is an PRG that fools f_a with error ε, then G can be used to simulate A without changing its acceptance probability by more than ε:

$$|\Pr[A(a, U_n) = 1] - \Pr[A(a, G(U_s)) = 1]| \leq \varepsilon.$$

Thus, if we wish to design a PRG to simulate A, we should study the computational complexity of the functions f_a.

Simulating randomized polynomial-time algorithms

One important case is when A is a *polynomial-time* randomized algorithm, corresponding to the complexity class **BPP**. In this case, the following claim says that the functions f_a can be computed by *polynomial-size Boolean circuits*.[5]

Claim 1.2 (PRGs for circuits can be used to simulate **BPP**)**.** Let A be a randomized decision algorithm and let a be an input. Let n be the number of random bits that A uses on input a and define $f_a\colon \{0,1\}^n \to \{0,1\}$ by the rule $f_a(x) = A(a,x)$. Let T be the running time of A on input a and assume $T \geq |a|$. Then f_a can be computed by a Boolean circuit of size $\mathrm{poly}(T)$.[6]

Proof. The function $A(a,x)$ can be computed by a Boolean circuit of size $\mathrm{poly}(T)$ that reads both a and x [190]. When we fix the "a" portion of the input bits to arbitrary values, what remains is a circuit of size $\mathrm{poly}(T)$ operating on x. □

Claim 1.2 implies that if $G\colon \{0,1\}^s \to \{0,1\}^n$ fools circuits of size $\mathrm{poly}(T)$, then G can be used to simulate time-T randomized algorithms. The running time of this simulation is essentially T plus the running time of G, so for this application, the appropriate "explicitness" condition is that G can be computed quickly, e.g., in time $\mathrm{poly}(T)$. Unfortunately, as discussed previously, the challenge of designing explicit PRGs for general Boolean circuits is extremely difficult.

[5] Recall that a Boolean circuit is a network of AND, OR, and NOT gates.

[6] Again, we assume for simplicity that n and T are determined by a rather than varying based on the random bits (cf. Footnote 4).

Remark 1.2 (Nonuniformity)**.** The Boolean circuit model is a *nonuniform* model, i.e., each individual Boolean circuit operates on inputs of some fixed length. The reader might find it counterintuitive that we seek PRGs for circuits in order to simulate *uniform* randomized polynomial-time algorithms (i.e., the one randomized algorithm can handle inputs of any arbitrary length). The concept of *advice* might be helpful [141]. Recall that a family of polynomial-size circuits (one circuit for each input length) is equivalent to a polynomial-time algorithm with a polynomial amount of advice: data that is trustworthy but that depends only on the input length. In our setting, the input a to the polynomial-time algorithm A can be viewed as advice that A uses to try to distinguish between truly random bits and the output of a PRG. We want to simulate A correctly even on a *worst-case* input a, and hence we want a PRG that fools an adversarial polynomial-time observer with advice, i.e., a Boolean circuit.

Simulating randomized log-space algorithms

Another important case is when A is a *log-space* randomized algorithm, corresponding to the complexity class **BPL**. In this case, for each input a, the function f_a can be computed by a *polynomial-width standard-order read-once branching program* (ROBP), defined next.

Definition 1.5 (Standard-order read-once branching programs)**.** A length-n *standard-order read-once branching program* (standard-order ROBP) f consists of a directed layered multigraph with $n+1$ layers, $V_0, \ldots, V_n$. For every $i < n$, each vertex $v \in V_i$ has two outgoing edges leading to V_{i+1}, one labeled 0 and the other labeled 1. Vertices in V_n have zero outgoing edges. There is a designated "start vertex" $v_{\text{start}} \in V_0$. An input $x \in \{0,1\}^n$ selects a path $(v_0, v_1, \ldots, v_n)$ through the graph: the path starts at $v_0 = v_{\text{start}}$, and upon reaching a vertex $v_i \in V_i$, the bit x_{i+1} specifies which outgoing edge to use. There is a designated set of "accept vertices" $V_{\text{accept}} \subseteq V_n$, and $f(x) = 1$ if $v_n \in V_{\text{accept}}$ and $f(x) = 0$ otherwise. The *width* of the program is the maximum number of vertices in a single layer (see Figure 1.2).

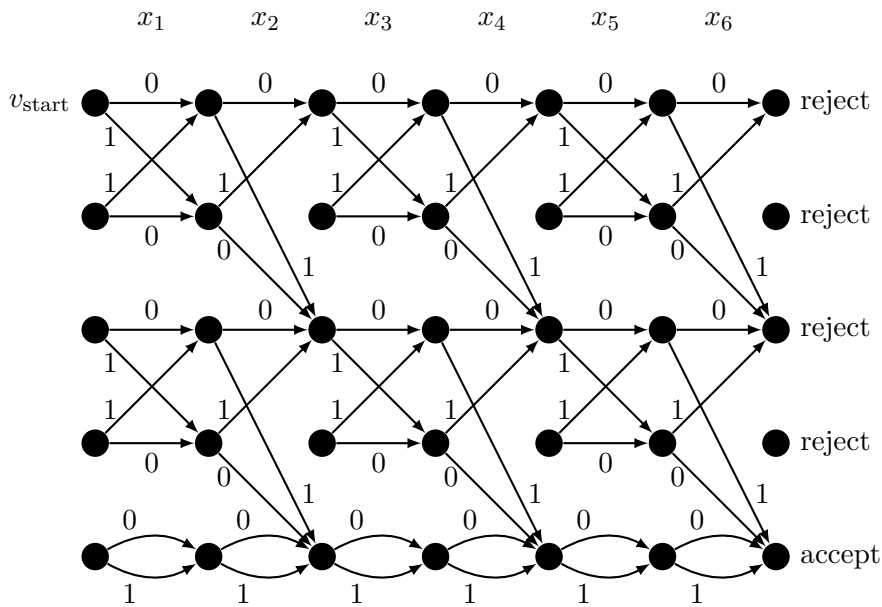

Figure 1.2: A width-5 length-6 standard-order ROBP computing the function $f(x) = \mathsf{MAJ}(x_1 \oplus x_2, x_3 \oplus x_4, x_5 \oplus x_6)$.

Claim 1.3 (PRGs for ROBPs can be used to simulate **BPL**)**.** Let A be a randomized decision algorithm and let a be an input. Let n be the number of random bits that A uses on input a and define $f_a \colon \{0,1\}^n \to \{0,1\}$ by the rule $f_a(x) = A(a,x)$. Let S be the number of bits of space used by A on input a and assume $S \geq \log |a|$. Then f_a can be computed by a standard-order ROBP of width $2^{O(S)}$.

Proof. We think of A as a Turing machine with an input tape, a work tape, and a random tape. Each vertex in the program corresponds to a configuration of A, consisting of the contents of its work tape, the location of the input tape and work tape read heads, and the internal state of A. An edge (u,v) labeled $b \in \{0,1\}$ indicates that if we run A on input a starting at configuration u until its next coin toss, and if that coin toss outcome is b, then the machine's configuration immediately following the coin toss is v. □

Remark 1.3 (The read-once property)**.** In general, a log-space algorithm with a polynomial amount of advice is equivalent to a polynomial-size branching program that might read its bits many times (see Definition 5.16). Nevertheless, we get a *read-once* branching program in Claim 1.3. The reason is that we are focusing on the behavior of the algorithm *as a function of its random bits*. An algorithm in the standard **BPL** model only has read-once access to its random tape: the algorithm cannot go back and re-read old random bits. (If one is computing using a single fair coin, then one cannot ask the coin what the outcome of the first toss was after tossing it a second time.)

Remark 1.4 (ROBP terminology)**.** In the pseudorandomness literature, standard-order ROBPs are often referred to as simply "ROBPs." This practice is a bit misleading, since the definition is *not* simply "a branching program that is read-once." Indeed, in addition to being read-once, we are assuming that the program is *oblivious*, meaning that the variable queried in time step i depends only on i, and more specifically, we are assuming that the branching program follows the *standard variable ordering*, meaning that in time step i, the program queries the variable x_i. (The branching program in the proof of Claim 1.3 indeed reads its input bits in the standard order, because without loss of generality, the algorithm A reads its read-once random tape from left to right.) Unsurprisingly, many papers outside the pseudorandomness literature use terms like "read-once branching program" to refer to more general models that are not necessarily even oblivious [17], [21], [22], [201], [247]. In this text, for clarity, we use the more verbose term "standard-order ROBP" to emphasize the variable ordering assumption.[7]

Claim 1.3 implies that if $G\colon \{0,1\}^s \to \{0,1\}^n$ fools standard-order ROBPs of width $2^{O(S)}$, then G can be used to simulate space-S randomized algorithms. For this application, the appropriate "explicitness" condition is that G can be computed in low space – perhaps space $O(S)$. More precisely, the space complexity of the deterministic simulation is essentially S plus the space complexity of computing $G(y)$ given *one-way read-only access* to the seed y.

[7] Hoza used the same verbose terminology in some other recent expository work [125].

Simulating other types of randomized algorithms

One can consider numerous other classes of randomized algorithms, as well as specific important randomized algorithms. In each case, if we wish to replace the truly random bits with pseudorandom bits, then the question we must answer is, what is the algorithm doing as a function of its random bits? If, for each fixed input a, the algorithm's behavior can be described by a function of "sufficiently low complexity" applied to its random bits, then we can design a PRG that fools such "low-complexity" functions and use it to simulate the algorithm. Because of the presence of the worst-case input a, the appropriate complexity measure will generally be captured by some *nonuniform* model of computation.

1.5.2 Derandomizing algorithms

If we use a PRG to simulate a randomized algorithm in the most natural possible way (as discussed above), we are still using a small amount of randomness, namely the truly random seed of the PRG. However, in many cases it is possible to eliminate this small amount of randomness, leading to a completely deterministic simulation. The most straightforward way to do this is to exhaustively try all possible seeds.

Claim 1.4 (Trying all seeds and taking a majority vote)**.** Let A be a randomized decision algorithm, let a be an input, and let n be the number of random bits that A uses on input a.[8] Let $\varepsilon > 0$ and assume that A succeeds with probability greater than $1/2+\varepsilon$, i.e., there is some "correct answer" $b \in \{0,1\}$ such that

$$\Pr[A(a, U_n) = b] > 1/2 + \varepsilon.$$

Define $f_a\colon \{0,1\}^n \to \{0,1\}$ by the rule $f_a(x) = A(a,x)$. Let $G\colon \{0,1\}^s \to \{0,1\}^n$ be a PRG that ε-fools f_a. Then

$$\mathsf{MAJ}_{y\in\{0,1\}^s}(A(a, G(y))) = b.$$

Proof. First, suppose $b = 1$. The definition of fooling implies that

$$\mathbb{E}[A(a, G(U_s))] = \mathbb{E}[f(G(U_s))] \geq \mathbb{E}[f] - \varepsilon > 1/2 + \varepsilon - \varepsilon = 1/2.$$

[8] Again, we assume for simplicity that n is determined by a.

Therefore, $A(a, G(y)) = 1$ for a majority of seeds y. Now suppose instead that $b = 0$. The fact that G fools f_a with error ε implies that G also fools $1 - f_a$ with error ε, because for any distribution X, we have

$$|\mathbb{E}[1-f_a(X)]-\mathbb{E}[1-f_a]| = |1-\mathbb{E}[f_a(X)]-1+\mathbb{E}[f_a]| = |\mathbb{E}[f_a]-\mathbb{E}[f_a(X)]|.$$

Therefore, by our previous analysis applied to $1 - A(a, x)$, we see that $A(a, G(y)) = 0$ for a majority of seeds y. □

Claim 1.4 implies, for example, that if $G\colon \{0,1\}^s \to \{0,1\}^n$ fools standard-order ROBPs of width $2^{O(S)}$, then we can use it to *deterministically* simulate randomized space-S decision algorithms. The space complexity of this deterministic simulation is essentially S, plus s, plus the space complexity of computing $G(y)$. Thus, for this application, the appropriate "explicitness" condition is that G can be computed in low space – perhaps space $O(s)$. In particular, for this application, there is no significant benefit to constructing a PRG with space complexity $o(s)$, because in the end we are going to use s bits of space to iterate through all possible seeds anyway.

The standard nonconstructive argument (Proposition 1.1) implies that there exists a nonexplicit ε-PRG for width-w length-n standard-order ROBPs with seed length $O(\log(wn/\varepsilon))$. Furthermore, the standard definition of **BPL** implies that randomized log-space algorithms have polynomial running time, and hence they use at most polynomially many random bits. Consequently, if we can design a PRG for standard-order ROBPs with seed length $O(\log(wn/\varepsilon))$ and space complexity $O(\log(wn/\varepsilon))$, then it will follow that $\mathbf{L} = \mathbf{BPL}$. That is, such a PRG would imply that randomized algorithms have at most a constant-factor advantage over deterministic algorithms in terms of space complexity. This would be a profound conclusion about the intrinsic relationship between randomness and memory as computational resources.

So far, optimal constructions of explicit PRGs for ROBPs are not known, but we do have "pretty good" constructions (see, e.g., Section 3.2). Furthermore, there are many partial derandomization results known for space-bounded computation, building on the theory of PRGs for ROBPs (in nontrivial ways). For example, it has been shown that randomized space-S algorithms can be simulated deterministically in

space slightly less than $S^{3/2}$ [124], [206]. The challenge of constructing optimal PRGs for standard-order ROBPs is an exciting and central open problem in the study of unconditional PRGs.

Other applications

We have briefly discussed the most straightforward applications of PRGs, namely simulating randomized algorithms using little or no randomness. We now give a small sample of less straightforward applications.

- Ironically, it turns out that PRGs are sometimes useful for *designing randomized algorithms*. For example, PRGs for space-bounded computation are often used in the design of randomized streaming algorithms using a technique first introduced by Indyk [136].
- Unconditional PRGs for restricted models have applications to "hardness amplification within **NP**" [105], [121], [160].
- Unconditional PRGs for restricted models have applications in the area of "meta-complexity." It turns out that PRGs can be used to rule out certain types of "natural proofs" of strong circuit lower bounds [199] or to show that certain models of computation cannot solve the "Minimum Circuit Size Problem" [137]. For these applications, the "correct" definition of explicitness is that for each fixed seed $y \in \{0,1\}^s$, there is a small Boolean circuit C_y such that for every $i \in [n]$, we have $C_y(i) = G(y)_i$.

1.6 Beyond PRGs: Hitting Set Generators and More

For the sake of context, in this section we briefly describe some relaxations of the PRG definition. The main motivation behind studying these relaxations is that constructing PRGs is challenging. These "generalized PRGs" are sometimes easier to construct, and yet they suffice for some (but not all) of the applications of PRGs. We only give a short overview of these concepts, since our main focus is true PRGs.

The most well-known "generalized PRG" concept is a *hitting set generator* (HSG).

Definition 1.6 (HSGs)**.** Suppose $\mathcal{F}$ is a class of functions $f\colon \{0,1\}^n \to \{0,1\}$. An *$\varepsilon$-HSG* for $\mathcal{F}$ is a function $G\colon \{0,1\}^s \to \{0,1\}^n$ such that for every $f \in \mathcal{F}$, if $\mathbb{E}[f] \geq \varepsilon$, then there exists some x such that $f(G(x)) = 1$.

An HSG is a "one-sided PRG." HSGs have been studied since the 1980s [3] if not earlier. HSGs can be used to derandomize algorithms that have one-sided error, simply by trying all seeds. In some contexts, HSGs can also be used (in nontrivial ways) to derandomize algorithms that have two-sided error [11], [12], [40], [61], [100].

A few years ago, [37] introduced a different generalization of PRGs, called *weighted PRGs* (WPRGs).[9]

Definition 1.7 (WPRG)**.** Suppose $\mathcal{F}$ is a class of functions $f\colon \{0,1\}^n \to \mathbb{R}$. An *$\varepsilon$-WPRG* for $\mathcal{F}$ is a pair (G, ρ), where $G\colon \{0,1\}^s \to \{0,1\}^n$ and $\rho\colon \{0,1\}^s \to \mathbb{R}$, such that for every $f \in \mathcal{F}$, we have

$$\left| \mathop{\mathbb{E}}_{U \sim U_s} [f(G(U)) \cdot \rho(U)] - \mathbb{E}[f] \right| \leq \varepsilon.$$

Thus, WPRGs generalize PRGs because we consider sparse *linear* combinations of the outputs of f rather than sparse *convex* combinations of the outputs of f. Several recent works have exploited this extra flexibility to construct WPRGs with better parameters than known PRGs [37], [52], [70], [124], [193].

Yet another generalization of PRGs is the concept of a *deterministic sampler*.

Definition 1.8 (Deterministic sampler)**.** Suppose $\mathcal{F}$ is a class of functions $f\colon \{0,1\}^n \to \mathbb{R}$. An *$\varepsilon$-deterministic sampler* for $\mathcal{F}$ is a deterministic oracle algorithm A that makes queries to a function $f \in \mathcal{F}$ and outputs a number $A^f \in \mathbb{R}$ such that $|A^f - \mathbb{E}[f]| \leq \varepsilon$.

The deterministic sampler model isolates a key feature of PRGs, which is that they are useful even if we merely have black-box access to the function f. Deterministic samplers have been discussed (by name)

[9] In Braverman, Cohen, and Garg's [37] original paper, they speak of "pseudorandom pseudo-distributions." The "weighted PRG" terminology was introduced later, by Cohen *et al.* [70].

in a few recent works [61], [191], [194]. Several older algorithms can also be understood as deterministic samplers [11], [12], [40], [100], [132].

One can show that these four concepts form a hierarchy:

$$\text{PRG} \implies \text{WPRG} \implies \text{deterministic sampler} \implies \text{HSG}.$$

Thus, PRGs (our focus in this text) are the most desirable of the four.

Appendices

A

Converse of the Sandwiching Lemma

Suppose we wish to show that every distribution that fools one class $\mathcal{F}_{\text{simp}}$ also fools another class $\mathcal{F}$. We presented two techniques for proving such a "transfer theorem":

1. The first technique is to express each $f \in \mathcal{F}$ as a linear combination of functions in $\mathcal{F}_{\text{simp}}$ and invoke the Triangle Inequality for PRG Errors.

2. The second technique is to sandwich each $f \in \mathcal{F}$ between functions in $\mathcal{F}_{\text{simp}}$ and invoke the Sandwiching Lemma.

As discussed in Section 2.5.1, we will now prove the following converse statement: If every distribution that fools $\mathcal{F}_{\text{simp}}$ also fools $\mathcal{F}$, then every $f \in \mathcal{F}$ is sandwiched between linear combinations of functions in $\mathcal{F}_{\text{simp}}$.

Theorem A.1 (Characterization of when fooling one class implies fooling another)**.** Let $n \in \mathbb{N}$, let $\mathcal{F}_{\text{simp}}$ be a finite class of functions $f\colon \{0,1\}^n \to \mathbb{R}$, and let $g\colon \{0,1\}^n \to \mathbb{R}$. Let $\varepsilon_0, \varepsilon > 0$ and suppose that every distribution X that fools $\mathcal{F}_{\text{simp}}$ with error ε_0 also fools g with error ε.

Then g is (2ε)-sandwiched between two functions $f_\ell, f_u \colon \{0,1\}^n \to \mathbb{R}$ of the form

$$f_\ell(x) = \lambda_\ell^{(0)} + \sum_{i=1}^{k_\ell} \lambda_\ell^{(i)} f_\ell^{(i)}(x) \tag{A.1}$$

$$f_u(x) = \lambda_u^{(0)} + \sum_{i=1}^{k_u} \lambda_u^{(i)} f_u^{(i)}(x), \tag{A.2}$$

where $k_\ell, k_u \in \mathbb{N}$, $\lambda_\ell^{(i)}, \lambda_u^{(i)} \in \mathbb{R}$, $f_\ell^{(i)}, f_u^{(i)} \in \mathcal{F}_{\text{simp}}$, and

$$\varepsilon_0 \cdot \sum_{i=1}^{k_\ell} |\lambda_\ell^{(i)}| \leq \varepsilon \tag{A.3}$$

$$\varepsilon_0 \cdot \sum_{i=1}^{k_u} |\lambda_u^{(i)}| \leq \varepsilon. \tag{A.4}$$

Conversely, if we start from the assumption that Equations (A.1) to (A.4) hold, then for any distribution X that fools $\mathcal{F}_{\text{simp}}$ with error ε_0, the Triangle Inequality for PRG Errors implies that X fools f_ℓ and f_u with error ε, and therefore the Sandwiching Lemma implies that X fools g with error 3ε. This recovers the assumption of Theorem A.1 up to a factor of three[1] in the error parameter. In this sense, Theorem A.1 shows that the Triangle Inequality for PRG Errors and the Sandwiching Lemma are "complete."

Before presenting the proof, let us elaborate on what the theorem says in two important special cases.

- Let $\mathcal{F}_{\text{simp}}$ be the class of Boolean k-juntas and let $\varepsilon_0 = 0$. Then Theorem A.1 says that a function is fooled by every k-wise uniform distribution if and only if the function can be sandwiched between two *low-degree real polynomials*. This was first shown by Bazzi [20] and, independently, by Benjamini *et al.* [26].

- Next, let $\mathcal{F}_{\text{simp}}$ to be the class of parity functions. Then Theorem A.1 essentially says that a function is fooled by every small-bias distribution if and only if the function can be sandwiched

[1] A more refined analysis, involving a more cumbersome version of the Sandwiching Lemma, gives a tight characterization without the extra factor of three.

between two functions with *low Fourier L_1 norm.*[2] This was first shown by De *et al.* [75].[3]

The general case seems to be folklore.

Proof of Theorem A.1. The proof uses linear programming duality. For each $f \in \mathcal{F}_{\text{simp}}$, define $\overline{f}\colon \{0,1\}^n \to \mathbb{R}$ by $\overline{f}(x) = f(x) - \mathbb{E}[f]$. Consider the following linear program in the variables $\{p_x\}_{x\in\{0,1\}^n}$:

$$\begin{aligned}
&\text{Maximize} \sum_{x\in\{0,1\}^n} p_x g(x),\\
&\text{subject to } p_x \geq 0 \text{ for all } x \in \{0,1\}^n\\
&\qquad \text{and} \sum_{x\in\{0,1\}^n} p_x = 1\\
&\qquad \text{and} \sum_{x\in\{0,1\}^n} p_x \overline{f}(x) \leq \varepsilon_0 \text{ for all } f \in \mathcal{F}_{\text{simp}}\\
&\qquad \text{and} -\sum_{x\in\{0,1\}^n} p_x \overline{f}(x) \leq \varepsilon_0 \text{ for all } f \in \mathcal{F}_{\text{simp}}.
\end{aligned}$$

The constraints say that the p_x variables are the probability mass function of some distribution that fools $\mathcal{F}_{\text{simp}}$ with error ε_0. The program is feasible, because if nothing else we can set $p_x = 2^{-n}$ (the uniform distribution). The objective function is the expectation of g under the distribution defined by the p_x variables, so the optimal value must be at most $\mathbb{E}[g] + \varepsilon$.

The dual linear program, in the variables z and $\{y_f^+, y_f^-\}_{f\in\mathcal{F}_{\text{simp}}}$, is as follows:

$$\begin{aligned}
&\text{Minimize } z + \varepsilon_0 \cdot \sum_{f\in\mathcal{F}_0} (y_f^+ + y_f^-),\\
&\text{subject to } y_f^+, y_f^- \geq 0 \text{ for all } f \in \mathcal{F}_{\text{simp}}\\
&\qquad \text{and } z + \sum_{f\in\mathcal{F}_0} \overline{f}(x) \cdot (y_f^+ - y_f^-) \geq g(x) \text{ for all } x \in \{0,1\}^n.
\end{aligned}$$

[2] Actually the quantity that matters is the sum of absolute values of the *nonempty* Fourier coefficients, whereas we included the empty Fourier coefficient in our definition of Fourier L_1 norm.

[3] Note that there is a minor mistake in the formulation by De *et al.* [75]: in their Proposition 2.7, the lower and upper sandwichers should be allowed to have different values of "l" and "δ."

By strong LP duality, the optimal value of this dual linear program is also at most $\mathbb{E}[g] + \varepsilon$. Observe that given a feasible solution to the dual linear program, if we subtract $\min\{y_f^+, y_f^-\}$ from y_f^+ and from y_f^-, then we get another feasible solution and the objective function can only decrease. Therefore, by setting $y_f = y_f^+ - y_f^-$, we obtain real numbers z^* and $\{y_f^*\}_{f \in \mathcal{F}_{\text{simp}}}$ such that

$$z^* + \varepsilon_0 \cdot \sum_{f \in \mathcal{F}_{\text{simp}}} |y_f^*| \leq \mathbb{E}[g] + \varepsilon, \text{ and}$$
$$z^* + \sum_{f \in \mathcal{F}_{\text{simp}}} \overline{f}(x) y_f^* \geq g(x) \text{ for all } x \in \{0,1\}^n.$$

Define

$$\begin{aligned} f_u(x) &= z^* + \sum_{f \in \mathcal{F}_{\text{simp}}} y_f^* \cdot \overline{f}(x) \\ &= \left(z^* - \sum_{f \in \mathcal{F}_{\text{simp}}} y_f^* \, \mathbb{E}[f] \right) + \sum_{f \in \mathcal{F}_{\text{simp}}} y_f^* \cdot f(x). \end{aligned}$$

Then f_u has the form given by Equation (A.2), and $f_u \geq g$. Furthermore, $\mathbb{E}[f_u] = z^*$, so

$$0 \leq \mathbb{E}[f_u - g] = z_* - \mathbb{E}[g] \leq \varepsilon - \varepsilon_0 \cdot \sum_{f \in \mathcal{F}_{\text{simp}}} |y_f^*|.$$

This shows that $\mathbb{E}[f_u - g] \leq \varepsilon$ and that Equation (A.4) holds.

Fooling g is equivalent to fooling $-g$, so the above also shows that there is some function f_ℓ of the form given by Equation (A.1) such that $-f_\ell \geq -g$, $\mathbb{E}[g - f_\ell] \leq \varepsilon$, and Equation (A.3) holds. Therefore, g is (2ε)-sandwiched between f_ℓ and f_u. □

B

List of PRGs

For reference, we conclude this text by listing the best explicit PRG constructions currently known for various models of computation, arranged by the model they fool. The list is not meant to be exhaustive; only a selection of important computational models are included. In each case, we only record a single state-of-the-art seed length, which in many cases is superior to the PRG constructions that we presented.

B.1 Circuit Models

In the list below, we use d to denote depth and m to denote size. Assume $d = O(1)$ and $m \geq n$.

- Conjunctions/disjunctions of literals
  - Seed length: $O(\log(1/\varepsilon) + \log\log n)$
  - Approach: k-wise δ-bias
  - Reference: Folklore
- $\mathbf{AC}^0$ circuits
  - Seed length: $\widetilde{O}(\log^{d-1} m \cdot \log(m/\varepsilon))$
  - Approach: Variant of the Ajtai-Wigderson framework
  - Reference: [166]
- Read-once CNFs/DNFs
  - Seed length: $O(\log n) + \widetilde{O}(\log(1/\varepsilon))$
  - Approach: Iterated restrictions with early termination
  - Reference: [81]
- Read-once $\mathbf{AC}^0$ formulas
  - Seed length: $\widetilde{O}(\log(n/\varepsilon))$
  - Approach: Iterated restrictions with early termination
  - References: [80], [82]
- De Morgan formulas
  - Seed length: $m^{1/3+o(1)} \cdot \mathrm{polylog}(1/\varepsilon)$
  - Approach: Variant of the IMZ framework
  - Reference: [120]
- Read-once De Morgan formulas
  - Seed length: $O(\log^2 n \cdot \log(n/\varepsilon))$
  - Approach: Iterated restrictions
  - Reference: [91]

B.2 Branching Program Models

In the list below, we use m to denote size and w to denote width. Assume $m \geq n$.

- Unrestricted branching programs
  - Seed length: $\sqrt{m} \cdot \text{polylog}(n/\varepsilon)$
  - Approach: Variant of the IMZ framework
  - Reference: [120]
- Width-2 branching programs that read d bits at a time
  - Seed length: $O(d \log n + d \cdot 2^d \cdot \log(m/\varepsilon))$
  - Approach: Sum of d δ-biased distributions
  - Reference: [29]
- Standard-order ROBPs with $w = 3$
  - Seed length: $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$
  - Approach: Iterated restrictions with early termination
  - Reference: [171]
- Standard-order ROBPs with $4 \leq w \leq n$
  - Seed length: $O(\log(n/\varepsilon) \cdot \log n)$
  - Approach: Recycling seeds
  - References: [131], [181]
- Standard-order ROBPs with $w \gg n$
  - Seed length: $O\left(\frac{\log(w/\varepsilon) \cdot \log n}{\log \log w}\right)$
  - Approach: Recycling seeds
  - References: [13], [140]

- Standard-order regular ROBPs
  - Seed length: $\widetilde{O}(\log(w/\varepsilon) \cdot \log n)$
  - Approach: INW generator
  - Reference: [38]
- Standard-order permutation ROBPs with $w = O(1)$
  - Seed length: $O(\log n \cdot \log(1/\varepsilon))$
  - Approach: INW generator
  - References: [74], [145], [224]
- Arbitrary-order ROBPs
  - Seed length: $O(\log(wn/\varepsilon) \cdot \log^2 n)$
  - Approach: Iterated restrictions
  - Reference: [91]
- Arbitrary-order ROBPs with $w = O(1)$
  - Seed length: $\widetilde{O}(\log(n/\varepsilon) \cdot \log n)$
  - Approach: Iterated restrictions
  - Reference: [91]
- Arbitrary-order permutation ROBPs with $w = O(1)$
  - Seed length: $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$
  - Approach: Polarizing random walks
  - Reference: [49]
- Decision trees, or more generally parity decision trees
  - Seed length: $O(\log(m/\varepsilon))$
  - Approach: δ-bias
  - Reference: [146]

B.3 Algebraic Models

- Parity functions
  - Seed length: $O(\log(n/\varepsilon))$
  - Approach: Balanced codes
  - References: [178], [217]
- Parities of at most k bits
  - Seed length: $O(\log(k/\varepsilon)) + \log\log n$
  - Approach: ε-biased seed for k-wise uniform generator
  - Reference: [178]
- Degree-d polynomials over $\mathbb{F}_2$
  - Seed length: $O(d\log n + d2^d \log(1/\varepsilon))$
  - Approach: Sum of d δ-biased distributions
  - Reference: [242]

B.4 Models Based on Locality

- $[-1, 1]$-valued k-juntas
  - Seed length: $O(k + \log(1/\varepsilon) + \log\log n)$
  - Approach: k-wise δ-bias
  - Reference: [178]
- Two-dimensional combinatorial rectangles
  - Seed length: $\frac{n}{2} + O(\log(1/\varepsilon))$
  - Approach: Random edge of expander
  - Reference: [131]
- d-dimensional combinatorial rectangles
  - Seed length: $\widetilde{O}(n/d + \log(1/\varepsilon) + \log\log n)$
  - Approach: Iterative alphabet reduction
  - Reference: [106]

- Two-party communication protocols with cost m
  - Seed length: $\frac{n}{2} + O(m + \log(1/\varepsilon))$
  - Approach: Random edge of expander
  - Reference: [131]

References

[1] S. Aaronson, "BQP and the polynomial hierarchy," in *Proc. 42nd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 141–150, 2010. DOI: 10.1145/1806689.1806711.

[2] A. Ahmadinejad, J. Kelner, J. Murtagh, J. Peebles, A. Sidford, and S. Vadhan, "High-precision estimation of random walks in small space," in *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 1295–1306, 2020. DOI: 10.1109/FOCS46700.2020.00123.

[3] M. Ajtai, J. Komlós, and E. Szemerédi, "Deterministic simulation in logspace," in *Proc. 19th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 132–140, 1987. DOI: 10.1145/28395.28410.

[4] M. Ajtai and A. Wigderson, "Deterministic simulation of probabilistic constant-depth circuits," *Advances in Computing Research – Randomness and Computation*, vol. 5, 1989, pp. 199–23.

[5] W. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr, "RSA and Rabin functions: Certain parts are as hard as the whole," *SIAM J. Comput.*, vol. 17, no. 2, 1988, pp. 194–209. DOI: 10.1137/0217013.

[6] N. Alon, "Eigenvalues and expanders," *Combinatorica*, vol. 6, no. 2, 1986, pp. 83–96. DOI: 10.1007/BF02579166.

[7] N. Alon, "Explicit expanders of every degree and size," *Combinatorica*, vol. 41, no. 4, 2021, pp. 447–463. DOI: 10.1007/s00493-020-4429-x.

[8] N. Alon, L. Babai, and A. Itai, "A fast and simple randomized parallel algorithm for the maximal independent set problem," *J. Algorithms*, vol. 7, no. 4, 1986, pp. 567–583. DOI: 10.1016/0196-6774(86)90019-2.

[9] N. Alon, O. Goldreich, J. Håstad, and R. Peralta, "Simple constructions of almost k-wise independent random variables," *Random Structures & Algorithms*, vol. 3, no. 3, 1992, pp. 289–304. DOI: 10.1002/rsa.3240030308.

[10] E. Anand and C. Umans, "Pseudorandomness of the sticky random walk," *arXiv preprint arXiv:2307.11104*, 2023.

[11] A. E. Andreev, A. E. F. Clementi, and J. D. P. Rolim, "A new general derandomization method," *J. ACM*, vol. 45, no. 1, 1998, pp. 179–213. DOI: 10.1145/273865.273933.

[12] A. E. Andreev, A. E. F. Clementi, J. D. P. Rolim, and L. Trevisan, "Weak random sources, hitting sets, and BPP simulations," *SIAM J. Comput.*, vol. 28, no. 6, 1999, pp. 2103–2116. DOI: 10.1137/S0097539797325636.

[13] R. Armoni, "On the derandomization of space-bounded computations," in *Proc. 2nd International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pp. 47–59, 1998. DOI: 10.1007/3-540-49543-6_5.

[14] R. Armoni, M. Saks, A. Wigderson, and S. Zhou, "Discrepancy sets and pseudorandom generators for combinatorial rectangles," in *Proc. 37th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 412–421, 1996. DOI: 10.1109/SFCS.1996.548500.

[15] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009. DOI: 10.1017/CBO9780511804090.

[16] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson, "BPP has subexponential time simulations unless EXPTIME has publishable proofs," *Comput. Complexity*, vol. 3, no. 4, 1993, pp. 307–318. DOI: 10.1007/BF01275486.

[17] L. Babai, P. Hajnal, E. Szemerédi, and G. Turán, "A lower bound for read-once-only branching programs," *J. Comput. System Sci.*, vol. 35, no. 2, 1987, pp. 153–162. DOI: 10.1016/0022-0000(87)90010-9.

[18] L. Babai, N. Nisan, and M. Szegedy, "Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs," *J. Comput. System Sci.*, vol. 45, no. 2, 1992, pp. 204–232. DOI: 10.1016/0022-0000(92)90047-M.

[19] D. A. Barrington, "Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1," *J. Comput. System Sci.*, vol. 38, no. 1, 1989, pp. 150–164. DOI: 10.1016/0022-0000(89)90037-8.

[20] L. M. J. Bazzi, "Polylogarithmic independence can fool DNF formulas," *SIAM J. Comput.*, vol. 38, no. 6, 2009, pp. 2220–2272. DOI: 10.1137/070691954.

[21] P. Beame, T. S. Jayram, and M. Saks, "Time-space tradeoffs for branching programs," *J. Comput. System Sci.*, vol. 63, no. 4, 2001, pp. 542–572. DOI: 10.1006/jcss.2001.1778.

[22] P. Beame, V. Liew, and M. Pătraşcu, "Finding the median (obliviously) with bounded space," in *Proc. 42nd International Colloquium on Automata, Languages and Programming (ICALP)*, pp. 103–115, 2015. DOI: 10.1007/978-3-662-47672-7_9.

[23] R. Beigel, N. Reingold, and D. A. Spielman, "The perceptron strikes back," in *Proc. 6th Annual IEEE Conference on Structure in Complexity Theory*, pp. 286, 287, 288, 289, 290, 291, Jul. 1991. DOI: 10.1109/SCT.1991.160270.

[24] M. Bellare and J. Rompel, "Randomness-efficient oblivious sampling," in *Proc. 35th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 276–287, 1994. DOI: 10.1109/SFCS.1994.365687.

[25] A. Ben-Aroya and A. Ta-Shma, "A combinatorial construction of almost-Ramanujan graphs using the zig-zag product," *SIAM J. Comput.*, vol. 40, no. 2, 2011, pp. 267–290. DOI: 10.1137/080732651.

[26] I. Benjamini, O. Gurel-Gurevich, and R. Peled, "On k-wise independent distributions and boolean functions," *arXiv:1201.3261*, 2012.

[27] L. Blum, M. Blum, and M. Shub, "A simple unpredictable pseudorandom number generator," *SIAM J. Comput.*, vol. 15, no. 2, 1986, pp. 364–383. DOI: 10.1137/0215025.

[28] M. Blum and S. Micali, "How to generate cryptographically strong sequences of pseudorandom bits," *SIAM J. Comput.*, vol. 13, no. 4, 1984, pp. 850–864. DOI: 10.1137/0213053.

[29] A. Bogdanov, Z. Dvir, E. Verbin, and A. Yehudayoff, "Pseudorandomness for width-2 branching programs," *Theory of Computing*, vol. 9, 2013, pp. 283–293. DOI: 10.4086/toc.2013.v009a007.

[30] A. Bogdanov, W. M. Hoza, G. Prakriya, and E. Pyne, "Hitting Sets for Regular Branching Programs," in *Proc. 37th Computational Complexity Conference (CCC)*, 3:1–3:22, 2022. DOI: 10.4230/LIPIcs.CCC.2022.3.

[31] A. Bogdanov, P. A. Papakonstantinou, and A. Wan, "Pseudorandomness for read-once formulas," in *FOCS*, R. Ostrovsky, Ed., pp. 240–246, IEEE, 2011.

[32] A. Bogdanov and E. Viola, "Pseudorandom bits for polynomials," *SIAM J. Comput.*, vol. 39, no. 6, 2010, pp. 2464–2486. DOI: 10.1137/070712109.

[33] R. B. Boppana, "The average sensitivity of bounded-depth circuits," *Inf. Process. Lett.*, vol. 63, no. 5, 1997, pp. 257–261. DOI: 10.1016/S0020-0190(97)00131-2.

[34] C. Bordenave, "A new proof of Friedman's second eigenvalue theorem and its extension to random lifts," *Ann. Sci. Éc. Norm. Supér. (4)*, vol. 53, no. 6, 2020, pp. 1393–1439. DOI: 10.24033/asens.245.

[35] A. Borodin, D. Dolev, F. E. Fich, and W. Paul, "Bounds for width two branching programs," *SIAM J. Comput.*, vol. 15, no. 2, 1986, pp. 549–560. DOI: 10.1137/0215040.

[36] M. Braverman, "Polylogarithmic independence fools AC^0 circuits," *J. ACM*, vol. 57, no. 5, 2010, 28:1–28:10. DOI: 10.1145/1754399.1754401.

[37] M. Braverman, G. Cohen, and S. Garg, "Pseudorandom pseudo-distributions with near-optimal error for read-once branching programs," *SIAM J. Comput.*, vol. 49, no. 5, 2020, STOC18-242–STOC18-299. DOI: 10.1137/18M1197734.

[38] M. Braverman, A. Rao, R. Raz, and A. Yehudayoff, "Pseudorandom generators for regular branching programs," *SIAM J. Comput.*, vol. 43, no. 3, 2014, pp. 973–986. DOI: 10.1137/120875673.

[39] J. Brody and E. Verbin, "The coin problem and pseudorandomness for branching programs," in *Proc. 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 30–39, 2010. DOI: 10.1109/FOCS.2010.10.

[40] H. Buhrman and L. Fortnow, "One-sided versus two-sided error in probabilistic computation," in *Proc. 16th Symposium on Theoretical Aspects of Computer Science (STACS)*, pp. 100–109, 1999. DOI: 10.1007/3-540-49116-3_9.

[41] J.-Y. Cai, A. Nerurkar, and D. Sivakumar, "Hardness and hierarchy theorems for probabilistic quasi-polynomial time," in *Proc. 31st Annual ACM Symposium on Theory of Computing (STOC)*, pp. 726–735, 1999. DOI: 10.1145/301250.301444.

[42] M. L. Carmosino, R. Impagliazzo, and M. Sabin, "Fine-Grained Derandomization: From Problem-Centric to Resource-Centric Complexity," in *Proc. 45th International Colloquium on Automata, Languages and Programming (ICALP)*, 27:1–27:16, 2018. DOI: 10.4230/LIPIcs.ICALP.2018.27.

[43] L. E. Celis, O. Reingold, G. Segev, and U. Wieder, "Balls and bins: Smaller hash families and faster evaluation," *SIAM J. Comput.*, vol. 42, no. 3, 2013, pp. 1030–1050. DOI: 10.1137/120871626.

[44] S. Chari, P. Rohatgi, and A. Srinivasan, "Improved algorithms via approximations of probability distributions," *J. Comput. System Sci.*, vol. 61, no. 1, 2000, pp. 81–107. DOI: 10.1006/jcss.1999.1695.

[45] E. Chattopadhyay, "Pseudorandomness and combinatorial constructions," 2018. URL: https://courses.cs.cornell.edu/cs6815/2018fa/.

[46] E. Chattopadhyay, "Pseudorandomness and combinatorial constructions," 2019. URL: https://courses.cs.cornell.edu/cs6815/2019fa/.

[47] E. Chattopadhyay, "Pseudorandomness and combinatorial constructions," 2022. URL: https://courses.cs.cornell.edu/cs6815/2022fa/.

[48] E. Chattopadhyay, J. Gaitonde, C. H. Lee, S. Lovett, and A. Shetty, "Fractional Pseudorandom Generators from Any Fourier Level," in *Proc. 36th Computational Complexity Conference (CCC)*, 10:1–10:24, 2021. DOI: 10.4230/LIPIcs.CCC.2021.10.

[49] E. Chattopadhyay, P. Hatami, K. Hosseini, and S. Lovett, "Pseudorandom generators from polarizing random walks," *Theory Comput.*, vol. 15, no. 1, 2019, pp. 1–26. DOI: 10.4086/toc.2019.v015a010.

[50] E. Chattopadhyay, P. Hatami, S. Lovett, and A. Tal, "Pseudorandom Generators from the Second Fourier Level and Applications to AC0 with Parity Gates," in *Proc. 10th Conference on Innovations in Theoretical Computer Science (ITCS)*, 22:1–22:15, 2018. DOI: 10.4230/LIPIcs.ITCS.2019.22.

[51] E. Chattopadhyay, P. Hatami, O. Reingold, and A. Tal, "Improved pseudorandomness for unordered branching programs through local monotonicity," in *Proc. 50th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 363–375, 2018. DOI: 10.1145/3188745.3188800.

[52] E. Chattopadhyay and J.-J. Liao, "Optimal error pseudodistributions for read-once branching programs," in *Proc. 35th Annual IEEE Conference on Computational Complexity (CCC)*, vol. 169, 25:1–25:27, 2020. DOI: 10.4230/LIPIcs.CCC.2020.25.

[53] E. Chattopadhyay and J.-J. Liao, *Recursive error reduction for regular branching programs*, ECCC preprint TR23-130, 2023. URL: https://eccc.weizmann.ac.il/report/2023/130/.

[54] L. Chen, W. M. Hoza, X. Lyu, A. Tal, and H. Wu, *Weighted pseudorandom generators via inverse analysis of random walks and shortcutting*, ECCC preprint TR23-114, 2023. URL: https://eccc.weizmann.ac.il/report/2023/114/.

[55] L. Chen, Z. Lu, X. Lyu, and I. C. Oliveira, "Majority vs. Approximate Linear Sum and Average-Case Complexity Below NC^1," in *Proc. 48th International Colloquium on Automata, Languages and Programming (ICALP)*, 51:1–51:20, 2021. DOI: 10.4230/LIPIcs.ICALP.2021.51.

[56] L. Chen, X. Lyu, A. Tal, and H. Wu, "New PRGs for Unbounded-Width/Adaptive-Order Read-Once Branching Programs," in *Proc. 50th International Colloquium on Automata, Languages and Programming (ICALP)*, vol. 261, 39:1–39:20, 2023. DOI: 10.4230/LIPIcs.ICALP.2023.39.

[57] L. Chen, X. Lyu, and R. R. Williams, "Almost-everywhere circuit lower bounds from non-trivial derandomization," in *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 1–12, 2020. DOI: 10.1109/FOCS46700.2020.00009.

[58] L. Chen and H. Ren, "Strong average-case circuit lower bounds from nontrivial derandomization," *SIAM J. Comput.*, vol. 51, no. 3, 2022, STOC20-115–STOC20-173. DOI: 10.1137/20M1364886.

[59] L. Chen, R. D. Rothblum, R. Tell, and E. Yogev, "On exponential-time hypotheses, derandomization, and circuit lower bounds: Extended abstract," in *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 13–23, 2020. DOI: 10.1109/FOCS46700.2020.00010.

[60] L. Chen and R. Tell, "Simple and fast derandomization from very hard functions: Eliminating randomness at almost no cost," in *Proc. 53rd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 283–291, 2021. DOI: 10.1145/3406325.3451059.

[61] K. Cheng and W. M. Hoza, "Hitting sets give two-sided derandomization of small space," *Theory of Computing*, vol. 18, no. 21, 2022, pp. 1–32. DOI: 10.4086/toc.2022.v018a021.

[62] K. Cheng and X. Li, "Efficient document exchange and error correcting codes with asymmetric information," in *Proc. 2021 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 2424–2443, 2021. DOI: 10.1137/1.9781611976465.144.

[63] M. Cheraghchi, V. Kabanets, Z. Lu, and D. Myrisiotis, "Circuit lower bounds for MCSP from local pseudorandom generators," *ACM Trans. Comput. Theory*, vol. 12, no. 3, 2020, Art. 21, 27. DOI: 10.1145/3404860.

[64] R. Chiclana and Y. Peres, "A local central limit theorem for random walks on expander graphs," *arXiv preprint arXiv:2212.00958*, 2022.

[65] B. Chor and O. Goldreich, "Unbiased bits from sources of weak randomness and probabilistic communication complexity," *SIAM J. on Computing*, vol. 17, no. 2, 1988, pp. 230–261. DOI: 10.1137/0217015.

[66] B. Chor, O. Goldreich, J. Håstad, J. Friedman, S. Rudich, and R. Smolensky, "The bit extraction problem or t-resilient functions," in *Proc. 26th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 396–407, 1985. DOI: 10.1109/SFCS.1985.55.

[67] S. M. Cioabă and M. R. Murty, "Expander graphs and gaps between primes," *Forum Math.*, vol. 20, no. 4, 2008, pp. 745–756. DOI: 10.1515/FORUM.2008.035.

[68] T. H. Click, A. Liu, and G. A. Kaminski, "Quality of random number generators significantly affects results of monte carlo simulations for organic and biological systems," *Journal of Computational Chemistry*, vol. 32, no. 3, 2011, pp. 513–524. DOI: 10.1002/jcc.21638.

[69] P. Coddington, "Analysis of random number generators using monte carlo simulation," *International Journal of Modern Physics C*, vol. 05, no. 03, 1994, pp. 547–560. DOI: 10.1142/S0129183194000726.

[70] G. Cohen, D. Doron, O. Renard, O. Sberlo, and A. Ta-Shma, "Error reduction for weighted PRGs against read once branching programs," in *Proc. 36th Computational Complexity Conference (CCC)*, 22:1–22:17, 2021. DOI: 10.4230/LIPIcs.CCC.2021.22.

[71] G. Cohen, D. Minzer, S. Peleg, A. Potechin, and A. Ta-Shma, "Expander Random Walks: The General Case and Limitations," in *Proc. 49th International Colloquium on Automata, Languages and Programming (ICALP)*, 43:1–43:18, 2022. DOI: 10.4230/LIPIcs.ICALP.2022.43.

[72] G. Cohen, N. Peri, and A. Ta-Shma, "Expander random walks: A Fourier-analytic approach," in *Proc. 53rd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 1643–1655, ACM, New York, 2021. DOI: 10.1145/3406325.3451049.

[73] M. B. Cohen, "Ramanujan graphs in polynomial time," in *Proc. 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 276–281, 2016. DOI: 10.1109/FOCS.2016.37.

[74] A. De, "Pseudorandomness for permutation and regular branching programs," in *Proc. 26th Annual IEEE Conference on Computational Complexity (CCC)*, pp. 221–231, 2011. DOI: 10.1109/CCC.2011.23.

[75] A. De, O. Etesami, L. Trevisan, and M. Tulsiani, "Improved pseudorandom generators for depth 2 circuits," in *Proc. 14th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pp. 504–517, 2010. DOI: 10.1007/978-3-642-15369-3_38.

[76] R. De Wolf, "A brief introduction to fourier analysis on the boolean cube," *Theory of Computing*, 2008, pp. 1–20.

[77] A. Degwekar, V. Vaikuntanathan, and P. N. Vasudevan, "Fine-grained cryptography," in *Proc. 36th Annual International Cryptology Conference (CRYPTO)*, pp. 533–562, 2016. DOI: 10.1007/978-3-662-53015-3_19.

[78] I. Diakonikolas, P. Gopalan, R. Jaiswal, R. A. Servedio, and E. Viola, "Bounded independence fools halfspaces," *SIAM J. Comput.*, vol. 39, no. 8, 2010, pp. 3441–3462. DOI: 10.1137/100783030.

[79] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM J. Comput.*, vol. 38, no. 1, 2008, pp. 97–139. DOI: 10.1137/060651380.

[80] D. Doron, P. Hatami, and W. M. Hoza, "Near-Optimal Pseudorandom Generators for Constant-Depth Read-Once Formulas," in *Proc. 34th Computational Complexity Conference (CCC)*, 16:1–16:34, 2019. DOI: 10.4230/LIPIcs.CCC.2019.16.

[81] D. Doron, P. Hatami, and W. M. Hoza, "Log-Seed Pseudorandom Generators via Iterated Restrictions," in *Proc. 35th Computational Complexity Conference (CCC)*, 6:1–6:36, 2020. DOI: 10.4230/LIPIcs.CCC.2020.6.

[82] D. Doron, R. Meka, O. Reingold, A. Tal, and S. Vadhan, "Pseudorandom Generators for Read-Once Monotone Branching Programs," in *Proc. 25th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, 58:1–58:21, 2021. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2021.58.

[83] D. Doron, D. Moshkovitz, J. Oh, and D. Zuckerman, "Nearly optimal pseudorandomness from hardness," in *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 1057–1068, 2020. DOI: 10.1109/FOCS46700.2020.00102.

[84] S. Egashira, Y. Wang, and K. Tanaka, "Fine-grained cryptography revisited," *J. Cryptology*, vol. 34, no. 3, 2021, Paper No. 23, 43. DOI: 10.1007/s00145-021-09390-3.

[85] P. Erdős, P. Frankl, and Z. Füredi, "Families of finite sets in which no set is covered by the union of r others," *Israel J. Math.*, vol. 51, no. 1-2, 1985, pp. 79–89. DOI: 10.1007/BF02772959.

[86] G. Even, O. Goldreich, M. Luby, N. Nisan, and B. Veličković, "Efficient approximation of product distributions," *Random Structures Algorithms*, vol. 13, no. 1, 1998, pp. 1–16. DOI: 10.1002/(SICI)1098-2418(199808)13:1<1::AID-RSA1>3.0.CO;2-W.

[87] B. Fefferman, R. Shaltiel, C. Umans, and E. Viola, "On beating the hybrid argument," *Theory Comput.*, vol. 9, 2013, pp. 809–843. DOI: 10.4086/toc.2013.v009a026.

[88] A. M. Ferrenberg, D. P. Landau, and Y. J. Wong, "Monte carlo simulations: Hidden errors from 'good' random number generators," *Phys. Rev. Lett.*, vol. 69, 23 Dec. 1992, pp. 3382–3384. DOI: 10.1103/PhysRevLett.69.3382.

[89] T. Filk, M. Marcu, and K. Fredenhagen, "Long range correlations in random number generators and their influence on monte carlo simulations," *Physics Letters B*, vol. 165, no. 1, 1985, pp. 125–130. DOI: 10.1016/0370-2693(85)90705-1.

[90] Y. Filmus, "Smolensky's lower bound," 2010. URL: https://yuvalfilmus.cs.technion.ac.il/Manuscripts/Smolensky.pdf.

[91] M. A. Forbes and Z. Kelley, "Pseudorandom generators for read-once branching programs, in any order," in *Proc. 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 946–955, 2018. DOI: 10.1109/FOCS.2018.00093.

[92] J. Friedman, "Some geometric aspects of graphs and their eigenfunctions," *Duke Math. J.*, vol. 69, no. 3, 1993, pp. 487–525. DOI: 10.1215/S0012-7094-93-06921-9.

[93] J. Friedman, "A proof of Alon's second eigenvalue conjecture and related problems," *Mem. Amer. Math. Soc.*, vol. 195, no. 910, 2008, pp. viii+100. DOI: 10.1090/memo/0910.

[94] Z. Füredi, "Matchings and covers in hypergraphs," *Graphs Combin.*, vol. 4, no. 2, 1988, pp. 115–206. DOI: 10.1007/BF01864160.

[95] O. Goldreich and L. A. Levin, "A hard-core predicate for all one-way functions," in *Proc. 21st Annual ACM Symposium on Theory of Computing (STOC)*, pp. 25–32, 1989. DOI: 10.1145/73007.73010.

[96] O. Goldreich, *Foundations of Cryptography Volume I: Basic Tools*. Cambridge University Press, 2001. DOI: 10.1017/CBO9780511546891.

[97] O. Goldreich, *A primer on pseudorandom generators*, vol. 55, ser. University Lecture Series. American Mathematical Society, Providence, RI, 2010, pp. x+114. DOI: 10.1090/ulect/055.

[98] O. Goldreich, "In a world of $\mathbf{P} = \mathbf{BPP}$," in *Studies in complexity and cryptography*, ser. Lecture Notes in Comput. Sci. Vol. 6650, Springer, Heidelberg, 2011, pp. 191–232. DOI: 10.1007/978-3-642-22670-0_20.

[99] O. Goldreich, H. Krawczyk, and M. Luby, "On the existence of pseudorandom generators," *SIAM J. Comput.*, vol. 22, no. 6, 1993, pp. 1163–1175. DOI: 10.1137/0222069.

[100] O. Goldreich, S. Vadhan, and A. Wigderson, "Simplified derandomization of BPP using a hitting set generator," in *Studies in Complexity and Cryptography*, ser. Lecture Notes in Computer Science, vol. 6650, Springer, Heidelberg, 2011, pp. 59–67. DOI: 10.1007/978-3-642-22670-0_8.

[101] S. Goldwasser, S. Micali, and P. Tong, "Why and how to establish a private code on a public network," in *Proc. 23rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 134–144, 1982. DOI: 10.1109/SFCS.1982.100.

[102] L. Golowich, "A new Berry-Esseen theorem for expander walks," in *Proc. 55th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 10–22, ACM, New York, 2023. DOI: 10.1145/3564246.3585141.

[103] L. Golowich and S. Vadhan, "Pseudorandomness of Expander Random Walks for Symmetric Functions and Permutation Branching Programs," in *Proc. 37th Computational Complexity Conference (CCC)*, 27:1–27:13, 2022. DOI: 10.4230/LIPIcs.CCC.2022.27.

[104] P. Gopalan, D. M. Kane, and R. Meka, "Pseudorandomness via the discrete Fourier transform," *SIAM J. Comput.*, vol. 47, no. 6, 2018, pp. 2451–2487. DOI: 10.1137/16M1062132.

[105] P. Gopalan, R. Meka, O. Reingold, L. Trevisan, and S. Vadhan, "Better pseudorandom generators from milder pseudorandom restrictions," in *Proc. 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 120–129, 2012. DOI: 10.1109/FOCS.2012.77.

[106] P. Gopalan and A. Yehudayoff, "Concentration for limited independence via inequalities for the elementary symmetric polynomials," *Theory Comput.*, vol. 16, 2020, Paper No. 17, 29. DOI: 10.4086/toc.2020.v016a017.

[107] P. Grassberger, "On correlations in 'good' random number generators," *Physics Letters A*, vol. 181, no. 1, 1993, pp. 43–46. DOI: 10.1016/0375-9601(93)91122-L.

[108] V. Guruswami and V. M. Kumar, "Pseudobinomiality of the Sticky Random Walk," in *Proc. 12th Conference on Innovations in Theoretical Computer Science (ITCS)*, vol. 185, 2021. DOI: 10.4230/LIPIcs.ITCS.2021.48.

[109] V. Guruswami, C. Umans, and S. Vadhan, "Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes," *J. ACM*, vol. 56, no. 4, 2009, Art. 20, 34. DOI: 10.1145/1538902.1538904.

[110] I. Haitner, D. Harnik, and O. Reingold, "Efficient pseudorandom generators from exponentially hard one-way functions," in *Proc. 33rd International Colloquium on Automata, Languages and Programming (ICALP)*, pp. 228–239, 2006. DOI: 10.1007/11787006_20.

[111] I. Haitner, D. Harnik, and O. Reingold, "On the power of the randomized iterate," *SIAM J. Comput.*, vol. 40, no. 6, 2011, pp. 1486–1528. DOI: 10.1137/080721820.

[112] I. Haitner, O. Reingold, and S. Vadhan, "Efficiency improvements in constructing pseudorandom generators from one-way functions," *SIAM J. Comput.*, vol. 42, no. 3, 2013, pp. 1405–1430. DOI: 10.1137/100814421.

[113] E. Haramaty, C. H. Lee, and E. Viola, "Bounded independence plus noise fools products," *SIAM J. Comput.*, vol. 47, no. 2, 2018, pp. 493–523. DOI: 10.1137/17M1129088.

[114] P. Harsha and S. Srinivasan, "On polynomial approximations to AC^0," *Random Structures Algorithms*, vol. 54, no. 2, 2019, pp. 289–303. DOI: 10.1002/rsa.20786.

[115] T. Hartman and R. Raz, "On the distribution of the number of roots of polynomials and explicit weak designs," *Random Structures Algorithms*, vol. 23, no. 3, 2003, pp. 235–263. DOI: 10.1002/rsa.10095.

[116] J. Hastad, "Almost optimal lower bounds for small depth circuits," *Adv. Comput. Res.*, vol. 5, 1989, pp. 143–170. URL: https://www.csc.kth.se/~johanh/largesmalldepth.pdf.

[117] J. Håstad, "A slight sharpening of lmn," *Journal of Computer and System Sciences*, vol. 63, no. 3, 2001, pp. 498–508. DOI: 10.1006/jcss.2001.1803.

[118] J. Håstad, "On the correlation of parity and small-depth circuits," *SIAM J. Comput.*, vol. 43, no. 5, 2014, pp. 1699–1708. DOI: 10.1137/120897432.

[119] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby, "A pseudorandom generator from any one-way function," *SIAM J. Comput.*, vol. 28, no. 4, 1999, pp. 1364–1396. DOI: 10.1137/S0097539793244708.

[120] P. Hatami, W. M. Hoza, A. Tal, and R. Tell, "Fooling constant-depth threshold circuits," in *Proc. 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 104–115, 2022. DOI: 10.1109/FOCS52979.2021.00019.

[121] A. Healy, S. Vadhan, and E. Viola, "Using nondeterminism to amplify hardness," *SIAM Journal on Computing*, vol. 35, no. 4, 2006, pp. 903–931. DOI: 10.1137/S0097539705447281.

[122] T. Holenstein, "Pseudorandom generators from one-way functions: A simple construction for any hardness," in *Theory of cryptography*, ser. Lecture Notes in Comput. Sci. Vol. 3876, Springer, Berlin, 2006, pp. 443–461. DOI: 10.1007/11681878_23.

[123] S. Hoory, N. Linial, and A. Wigderson, "Expander graphs and their applications," *Bull. Amer. Math. Soc. (N.S.)*, vol. 43, no. 4, 2006, pp. 439–561. DOI: 10.1090/S0273-0979-06-01126-8.

[124] W. M. Hoza, "Better pseudodistributions and derandomization for space-bounded computation," in *Proc. 25th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, 28:1–28:23, 2021. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2021.28.

[125] W. M. Hoza, "Recent progress on derandomizing space-bounded computation," *Bulletin of the EATCS*, no. 138, 2022, pp. 114–143. URL: https://eatcs.org/images/bulletin/beatcs138.pdf.

[126] W. M. Hoza, *A technique for hardness amplification against* AC^0, ECCC preprint TR23-176, 2023. URL: https://eccc.weizmann.ac.il/report/2023/176/.

[127] W. M. Hoza, E. Pyne, and S. Vadhan, "Pseudorandom Generators for Unbounded-Width Permutation Branching Programs," in *Proc. 12th Conference on Innovations in Theoretical Computer Science (ITCS)*, 7:1–7:20, 2021. DOI: 10.4230/LIPIcs.ITCS.2021.7.

[128] W. M. Hoza and D. Zuckerman, "Simple optimal hitting sets for small-success **RL**," *SIAM J. Comput.*, vol. 49, no. 4, 2020, pp. 811–820. DOI: 10.1137/19M1268707.

[129] R. Impagliazzo, W. Matthews, and R. Paturi, "A satisfiability algorithm for AC^0," in *Proc. 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 961–972, 2012. URL: https://dl.acm.org/doi/10.5555/2095116.2095193.

[130] R. Impagliazzo, R. Meka, and D. Zuckerman, "Pseudorandomness from shrinkage," *J. ACM*, vol. 66, no. 2, 2019, Art. 11, 16. DOI: 10.1145/3230630.

[131] R. Impagliazzo, N. Nisan, and A. Wigderson, "Pseudorandomness for network algorithms," in *Proc. 26th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 356–364, 1994. DOI: 10.1145/195058.195190.

[132] R. Impagliazzo, R. Shaltiel, and A. Wigderson, "Near-optimal conversion of hardness into pseudo-randomness," in *Proc. 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 181–190, 1999. DOI: 10.1109/SFFCS.1999.814590.

[133] R. Impagliazzo, R. Shaltiel, and A. Wigderson, "Reducing the seed length in the Nisan-Wigderson generator," *Combinatorica*, vol. 26, no. 6, 2006, pp. 647–681. DOI: 10.1007/s00493-006-0036-8.

[134] R. Impagliazzo and A. Wigderson, "P = BPP if E requires exponential circuits: Derandomizing the XOR lemma," in *Proc. 29th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 220–229, 1997. DOI: 10.1145/258533.258590.

[135] R. Impagliazzo and A. Wigderson, "Randomness vs time: Derandomization under a uniform assumption," *J. Comput. System Sci.*, vol. 63, no. 4, 2001, pp. 672–688. DOI: 10.1006/jcss.2001.1780.

[136] P. Indyk, "Stable distributions, pseudorandom generators, embeddings, and data stream computation," *J. ACM*, vol. 53, no. 3, 2006, pp. 307–323. DOI: 10.1145/1147954.1147955.

[137] V. Kabanets and J.-Y. Cai, "Circuit minimization problem," in *Proc. 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 73–79, 2000. DOI: 10.1145/335305.335314.

[138] V. Kabanets and Z. Lu, "Satisfiability and Derandomization for Small Polynomial Threshold Circuits," in *Proc. 22nd International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, 46:1–46:19, 2018. DOI: 10.4230/LIPIcs.APPROX-RANDOM.2018.46.

[139] C. Kalle and S. Wansleben, "Problems with the random number generator ranf implemented on the cdc cyber 205," *Computer Physics Communications*, vol. 33, no. 4, 1984, pp. 343–346. DOI: 10.1016/0010-4655(84)90139-5.

[140] D. M. Kane, J. Nelson, and D. P. Woodruff, *Revisiting norm estimation in data streams*, 2008. arXiv: 0811.3648 [cs.DS].

[141] R. M. Karp and R. J. Lipton, "Some connections between nonuniform and uniform complexity classes," in *Proc. 12th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 302–309, 1980. DOI: 10.1145/800141.804678.

[142] Z. Kelley, "An improved derandomization of the switching lemma," in *Proc. 53rd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 272–282, 2021. DOI: 10.1145/3406325.3451054.

[143] A. R. Klivans, H. K. Lee, and A. Wan, "Mansour's conjecture is true for random DNF formulas," in *Proc. 23rd Conference on Learning Theory (COLT)*, pp. 368–380, 2010. URL: http://www.learningtheory.org/colt2010/papers/085Lee.pdf.

[144] A. R. Klivans and D. van Melkebeek, "Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses," *SIAM J. Comput.*, vol. 31, no. 5, 2002, pp. 1501–1526. DOI: 10.1137/S0097539700389652.

[145] M. Koucký, P. Nimbhorkar, and P. Pudlák, "Pseudorandom generators for group products," in *Proc. 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 263–272, 2011. DOI: 10.1145/1993636.1993672.

[146] E. Kushilevitz and Y. Mansour, "Learning decision trees using the Fourier spectrum," *SIAM J. Comput.*, vol. 22, no. 6, 1993, pp. 1331–1348. DOI: 10.1137/0222080.

[147] P. L'Ecuyer and R. Simard, "Testu01: A c library for empirical testing of random number generators," *ACM Trans. Math. Softw.*, vol. 33, no. 4, Aug. 2007. DOI: 10.1145/1268776.1268777.

[148] C. H. Lee, "Fourier bounds and pseudorandom generators for product tests," in *Proc. 34th Computational Complexity Conference (CCC)*, 7:1–7:25, 2019. DOI: 10.4230/LIPIcs.CCC.2019.7.

[149] C. H. Lee, E. Pyne, and S. Vadhan, "Fourier Growth of Regular Branching Programs," in *Proc. 26th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, 2:1–2:21, 2022. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2022.2.

[150] C. H. Lee, E. Pyne, and S. Vadhan, "On the Power of Regular and Permutation Branching Programs," in *Proc. 27th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, 44:1–44:22, 2023. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2023.44.

[151] C. H. Lee and E. Viola, "Some limitations of the sum of small-bias distributions," *Theory Comput.*, vol. 13, 2017, Paper No. 16, 23. DOI: 10.4086/toc.2017.v013a016.

[152] C. H. Lee and E. Viola, "More on bounded independence plus noise: Pseudorandom generators for read-once polynomials," *Theory Comput.*, vol. 16, 2020, Paper No. 7, 50. DOI: 10.4086/toc.2020.v016a007.

[153] L. A. Levin, "One way functions and pseudorandom generators," *Combinatorica*, vol. 7, no. 4, 1987, pp. 357–363. DOI: 10.1007/BF02579323.

[154] N. Linial, M. Luby, M. Saks, and D. Zuckerman, "Efficient construction of a small hitting set for combinatorial rectangles in high dimension," *Combinatorica*, vol. 17, no. 2, 1997, pp. 215–234. DOI: 10.1007/BF01200907.

[155] N. Linial, Y. Mansour, and N. Nisan, "Constant depth circuits, Fourier transform, and learnability," *Journal of the ACM*, vol. 40, no. 3, 1993, pp. 607–620. DOI: 10.1145/174130.174138.

[156] N. Linial and N. Nisan, "Approximate inclusion-exclusion," *Combinatorica*, vol. 10, no. 4, 1990, pp. 349–365. DOI: 10.1007/BF02128670.

[157] S. Lovett, "Unconditional pseudorandom generators for low degree polynomials," *Theory of Computing*, vol. 5, no. 1, 2009, pp. 69–82. DOI: 10.4086/toc.2009.v005a003.

[158] S. Lovett and S. Srinivasan, "Correlation bounds for poly-size AC^0 circuits with $n^{1-o(1)}$ symmetric gates," in *Proc. 15th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pp. 640–651, 2011. DOI: 10.1007/978-3-642-22935-0_54.

[159] C.-J. Lu, "Improved pseudorandom generators for combinatorial rectangles," *Combinatorica*, vol. 22, no. 3, 2002, pp. 417–433. DOI: 10.1007/s004930200021.

[160] C.-J. Lu, S.-C. Tsai, and H.-L. Wu, "Improved hardness amplification in NP," *Theoret. Comput. Sci.*, vol. 370, no. 1-3, 2007, pp. 293–298. DOI: 10.1016/j.tcs.2006.10.009.

[161] A. Lubotzky, R. Phillips, and P. Sarnak, "Ramanujan graphs," *Combinatorica*, vol. 8, no. 3, 1988, pp. 261–277. DOI: 10.1007/BF02126799.

[162] A. Lubotzky, "Expander graphs in pure and applied mathematics," *Bull. Amer. Math. Soc. (N.S.)*, vol. 49, no. 1, 2012, pp. 113–162. DOI: 10.1090/S0273-0979-2011-01359-3.

[163] M. Luby and B. Veličković, "On deterministic approximation of DNF," *Algorithmica*, vol. 16, no. 4/5, 1996, pp. 415–433. DOI: 10.1007/BF01940873.

[164] M. Luby, B. Veličković, and A. Wigderson, "Deterministic approximate counting of depth-2 circuits," in *Proc. 2nd Israel Symposium on Theory and Computing Systems (ISTCS)*, pp. 18–24, 1993. DOI: 10.1109/ISTCS.1993.253488.

[165] M. Luby and A. Wigderson, "Pairwise independence and derandomization," *Foundations and Trends in Theoretical Computer Science*, vol. 1, no. 4, 2006, pp. 237–301. DOI: 10.1561/0400000009.

[166] X. Lyu, "Improved Pseudorandom Generators for AC^0 Circuits," in *Proc. 37th Computational Complexity Conference (CCC)*, 34:1–34:25, 2022. DOI: 10.4230/LIPIcs.CCC.2022.34.

[167] A. W. Marcus, D. A. Spielman, and N. Srivastava, "Interlacing families I: Bipartite Ramanujan graphs of all degrees," *Ann. of Math. (2)*, vol. 182, no. 1, 2015, pp. 307–325. DOI: 10.4007/annals.2015.182.1.7.

[168] G. A. Margulis, "Explicit group-theoretic constructions of combinatorial schemes and their applications in the construction of expanders and concentrators," *Problemy Peredachi Informatsii*, vol. 24, no. 1, 1988, pp. 51–60. URL: http://mi.mathnet.ru/eng/ppi/v24/i1/p51.

[169] M. Matsumoto and T. Nishimura, "Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Trans. Model. Comput. Simul.*, vol. 8, no. 1, Jan. 1998, pp. 3–30. DOI: 10.1145/272991.272995.

[170] N. Mazor and J. Zhang, "Simple constructions from (almost) regular one-way functions," in *Proc. 19th Theory of Cryptography Conference (TCC)*, pp. 457–485, 2021. DOI: 10.1007/978-3-030-90453-1_16.

[171] R. Meka, O. Reingold, and A. Tal, "Pseudorandom generators for width-3 branching programs," in *Proc. 51st Annual ACM Symposium on Theory of Computing (STOC)*, pp. 626–637, 2019. DOI: 10.1145/3313276.3316319.

[172] R. Meka and D. Zuckerman, "Small-bias spaces for group products," in *Proc. 13th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pp. 658–672, 2009. DOI: 10.1007/978-3-642-03685-9_49.

[173] R. Meka and D. Zuckerman, "Pseudorandom generators for polynomial threshold functions," *SIAM J. Comput.*, vol. 42, no. 3, 2013, pp. 1275–1301. DOI: 10.1137/100811623.

[174] A. Milchev, K. Binder, and D. Heermann, "Fluctuations and lack of self-averaging in the kinetics of domain growth," *Zeitschrift für Physik B Condensed Matter*, vol. 63, no. 4, 1986, pp. 521–535. DOI: 10.1007/BF01726202.

[175] P. B. Miltersen, "Derandomizing complexity classes," in *Handbook of randomized computing, Vol. I, II*, ser. Comb. Optim. Vol. 9, Kluwer Acad. Publ., Dordrecht, 2001, pp. 843–941. DOI: 10.1007/978-1-4615-0013-1_19.

[176] S. Mohanty, R. O'Donnell, and P. Paredes, "Explicit near-Ramanujan graphs of every degree," in *Proc. 52nd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 510–523, ACM, New York, 2020. DOI: 10.1145/3357713.3384231.

[177] M. Morgenstern, "Existence and explicit constructions of $q + 1$ regular Ramanujan graphs for every prime power q," *J. Combin. Theory Ser. B*, vol. 62, no. 1, 1994, pp. 44–62. DOI: 10.1006/jctb.1994.1054.

[178] J. Naor and M. Naor, "Small-bias probability spaces: Efficient constructions and applications," *SIAM J. Comput.*, vol. 22, no. 4, 1993, pp. 838–856. DOI: 10.1137/0222053.

[179] A. Nilli, "On the second eigenvalue of a graph," *Discrete Math.*, vol. 91, no. 2, 1991, pp. 207–210. DOI: 10.1016/0012-365X(91)90112-F.

[180] N. Nisan, "Pseudorandom bits for constant depth circuits," *Combinatorica*, vol. 11, no. 1, 1991, pp. 63–70. DOI: 10.1007/BF01375474.

[181] N. Nisan, "Pseudorandom generators for space-bounded computation," *Combinatorica*, vol. 12, no. 4, 1992, pp. 449–461. DOI: 10.1007/BF01305237.

[182] N. Nisan and A. Ta-Shma, "Extracting randomness: A survey and new constructions," *J. Comput. System Sci.*, vol. 58, no. 1, part 2, 1999, pp. 148–173. DOI: 10.1006/jcss.1997.1546.

[183] N. Nisan and A. Wigderson, "Hardness vs randomness," *J. Comput. Syst. Sci.*, vol. 49, no. 2, 1994, pp. 149–167. DOI: 10.1016/S0022-0000(05)80043-1.

[184] N. Nisan and D. Zuckerman, "Randomness is linear in space," *J. Comput. System Sci.*, vol. 52, no. 1, 1996, pp. 43–52. DOI: 10.1006/jcss.1996.0004.

[185] R. O'Donnell, *Analysis of Boolean Functions*. Cambridge University Press, 2014. DOI: 10.1017/CBO9781139814782.

[186] C. H. Papadimitriou and M. Sipser, "Communication complexity," *J. Comput. System Sci.*, vol. 28, no. 2, 1984, pp. 260–269. DOI: 10.1016/0022-0000(84)90069-2.

[187] G. Parisi and F. Rapuano, "Effects of the random number generator on computer simulations," *Physics Letters B*, vol. 157, no. 4, 1985, pp. 301–302. DOI: 10.1016/0370-2693(85)90670-7.

[188] R. Peralta, "On the randomness complexity of algorithms," *University of Wisconsin, Milwaukee CS Research Report TR 90-1*, 1990.

[189] N. Perlroth, "Government announces steps to restore confidence on encryption standards," *The New York Times*, 2013. URL: https://bits.blogs.nytimes.com/2013/09/10/government-announces-steps-to-restore-confidence-on-encryption-standards/ (accessed on 07/14/2021).

[190] N. Pippenger and M. J. Fischer, "Relations among complexity measures," *Journal of the ACM*, vol. 26, no. 2, 1979, pp. 361–381. DOI: 10.1145/322123.322138.

[191] E. Pyne, R. Raz, and W. Zhan, *Certified hardness vs. randomness for log-space*, ECCC preprint TR23-040, 2023. URL: https://eccc.weizmann.ac.il/report/2023/040/.

[192] E. Pyne and S. Vadhan, "Limitations of the Impagliazzo-Nisan-Wigderson pseudorandom generator against permutation branching programs," in *Proc. 27th International Computing and Combinatorics Conference (COCOON)*, pp. 3–12, 2021. DOI: 10.1007/978-3-030-89543-3_1.

[193] E. Pyne and S. Vadhan, "Pseudodistributions that beat all pseudorandom generators (extended abstract)," in *Proc. 36th Computational Complexity Conference (CCC)*, 33:1–33:15, 2021. DOI: 10.4230/LIPIcs.CCC.2021.33, Full version: ECCC preprint TR21-019.

[194] E. Pyne and S. Vadhan, "Deterministic approximation of random walks via queries in graphs of unbounded size," in *Proc. 5th Symposium on Simplicity in Algorithms (SOSA)*, pp. 57–67, 2022. DOI: 10.1137/1.9781611977066.5.

[195] Y. Rabani and A. Shpilka, "Explicit construction of a small ε-net for linear threshold functions," *SIAM J. on Computing*, vol. 39, no. 8, 2010, pp. 3501–3520. DOI: 10.1137/090764190.

[196] A. Rao and A. Yehudayoff, *Communication Complexity and Applications*. Cambridge University Press, 2020. DOI: 10.1017/9781108671644.

[197] R. Raz, O. Reingold, and S. Vadhan, "Extracting all the randomness and reducing the error in Trevisan's extractors," *J. Comput. System Sci.*, vol. 65, no. 1, 2002, pp. 97–128. DOI: 10.1006/jcss.2002.1824.

[198] A. A. Razborov, "Lower bounds on the size of bounded depth circuits over a complete basis with logical addition," *Math. Notes*, vol. 41, no. 4, 1987, pp. 333–338. DOI: 10.1007/BF01137685.

[199] A. A. Razborov and S. Rudich, "Natural proofs," *J. Comput. Syst. Sci.*, vol. 55, no. 1, 1997, pp. 24–35.

[200] A. Razborov, "A simple proof of Bazzi's theorem," *ACM Transactions on Computation Theory*, vol. 1, no. 1, 2009. DOI: 10.1145/1490270.1490273.

[201] A. A. Razborov, "Lower bounds for deterministic and nondeterministic branching programs," in *Proc. 8th International Conference on Fundamentals of Computation Theory (FCT)*, pp. 47–60, 1991. DOI: 10.1007/3-540-54458-5_49.

[202] O. Reingold, T. Steinke, and S. Vadhan, "Pseudorandomness for regular branching programs via Fourier analysis," in *Proc. 17th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pp. 655–670, 2013. DOI: 10.1007/978-3-642-40328-6_45.

[203] O. Reingold, L. Trevisan, and S. Vadhan, "Pseudorandom walks on regular digraphs and the **RL** vs. **L** problem," in *Proc. 38th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 457–466, 2006. DOI: 10.1145/1132516.1132583.

[204] V. Rödl, "On a packing and covering problem," *European Journal of Combinatorics*, vol. 6, no. 1, 1985, pp. 69–78. DOI: 10.1016/S0195-6698(85)80023-8.

[205] B. Rossman, "Criticality of Regular Formulas," in *Proc. 34th Computational Complexity Conference (CCC)*, 1:1–1:28, 2019. DOI: 10.4230/LIPIcs.CCC.2019.1.

[206] M. Saks and S. Zhou, "$\mathrm{BP_HSPACE}(S) \subseteq \mathrm{DSPACE}(S^{3/2})$," *J. Comput. System Sci.*, vol. 58, no. 2, 1999, pp. 376–403. DOI: 10.1006/jcss.1998.1616.

[207] J. Schönheim, "On coverings," *Pacific J. Math.*, vol. 14, 1964, pp. 1405–1411. URL: http://projecteuclid.org/euclid.pjm/1103033815.

[208] R. A. Servedio and L.-Y. Tan, "Luby-Veličković-Wigderson revisited: Improved correlation bounds and pseudorandom generators for depth-two circuits," in *Proc. 22nd International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, 56:1–56:20, 2018. DOI: 10.4230/LIPIcs.APPROX-RANDOM.2018.56.

[209] R. A. Servedio and L.-Y. Tan, "Improved Pseudorandom Generators from Pseudorandom Multi-Switching Lemmas," in *Proc. 28th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, 45:1–45:23, 2019. DOI: 10.4230/LIPIcs.APPROX-RANDOM.2019.45.

[210] R. A. Servedio and L.-Y. Tan, "Pseudorandomness for read-k DNF formulas," in *Proc. 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 621–638, 2019. DOI: 10.1137/1.9781611975482.39.

[211] R. Shaltiel, "Recent developments in extractors," *Bulletin of the European Association for Theoretical Computer Science*, vol. 77, Jun. 2002, pp. 67–95.

[212] R. Shaltiel, "An introduction to randomness extractors," in *Proc. 38th International Colloquium on Automata, Languages and Programming (ICALP)*, pp. 21–41, 2011. DOI: 10.1007/978-3-642-22012-8_2.

[213] R. Shaltiel and C. Umans, "Simple extractors for all min-entropies and a new pseudorandom generator.," *J. ACM*, vol. 52, no. 2, 2005, pp. 172–216. DOI: 10.1145/1059513.1059516.

[214] A. Shamir, "On the generation of cryptographically strong pseudorandom sequences," *ACM Trans. Comput. Syst.*, vol. 1, no. 1, 1983, pp. 38–44. DOI: 10.1145/357353.357357.

[215] A. Ta-Shma, "Randomized algorithms and de-randomization," 2015. URL: http://www.cs.tau.ac.il/~amnon/Classes/2015-PRG/class.htm.

[216] A. Ta-Shma, "Expanders, pseudorandomness and derandomization," 2016. URL: http://www.cs.tau.ac.il/~amnon/Classes/2016-PRG/class.htm.

[217] A. Ta-Shma, "Explicit, almost optimal, epsilon-balanced codes," in *Proc. 49th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 238–251, 2017. DOI: 10.1145/3055399.3055408.

[218] A. Ta-Shma, "Space-bounded computation," 2018. URL: http://www.cs.tau.ac.il/~amnon/Classes/2018-Space/class.htm.

[219] A. Ta-Shma, "A first course in derandomization," 2019. URL: http://www.cs.tau.ac.il/~amnon/Classes/2019-Derandomization/class.htm.

[220] J. Šíma and S. Žák, "A polynomial-time construction of a hitting set for read-once branching programs of width 3," *Fund. Inform.*, vol. 184, no. 4, 2021, pp. 307–354. DOI: 10.3233/fi-2021-2101.

[221] M. Skorski, "Tight Chernoff-Like Bounds Under Limited Independence," in *Proc. 26th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, 15:1–15:14, 2022. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2022.15.

[222] R. Smolensky, "On representations by low-degree polynomials," in *Proc. 34th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 130–138, 1993. DOI: 10.1109/SFCS.1993.366874.

[223] R. Smolensky, "Algebraic methods in the theory of lower bounds for Boolean circuit complexity," in *Proc. 19th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 77–82, 1987. DOI: 10.1145/28395.28404.

[224] T. Steinke, *Pseudorandomness for permutation branching programs without the group theory*, ECCC preprint TR12-083, 2012. URL: https://eccc.weizmann.ac.il/report/2012/083/.

[225] T. Steinke, S. Vadhan, and A. Wan, "Pseudorandomness and Fourier-growth bounds for width-3 branching programs," *Theory Comput.*, vol. 13, 2017, Paper No. 12. DOI: 10.4086/toc.2017.v013a012.

[226] B. A. Subbotovskaya, "Realizations of linear function by formulas using $+, \cdot, -$," *Doklady Akademii Nauk SSSR*, vol. 136:3, 1961, pp. 553–555. URL: http://mi.mathnet.ru/eng/dan/v136/i3/p553.

[227] M. Sudan, L. Trevisan, and S. Vadhan, "Pseudorandom generators without the xor lemma," *J. Comput. Syst. Sci.*, vol. 62, no. 2, 2001, pp. 236–266. DOI: 10.1006/jcss.2000.1730.

[228] A. Tal, "Tight Bounds on the Fourier Spectrum of AC0," in *Proc. 32nd Computational Complexity Conference (CCC)*, 15:1–15:31, 2017. DOI: 10.4230/LIPIcs.CCC.2017.15.

[229] A. Tal, "Pseudorandomness," 2021. URL: https://www.avishaytal.org/pseudorandomness.

[230] J. Tarui, "Probabilistic polynomials, AC^0 functions and the polynomial-time hierarchy," *Theoret. Comput. Sci.*, vol. 113, no. 1, 1993, pp. 167–183. DOI: 10.1016/0304-3975(93)90214-E.

[231] S. Toda and M. Ogiwara, "Counting classes are at least as hard as the polynomial-time hierarchy," *SIAM J. Comput.*, vol. 21, no. 2, 1992, pp. 316–328. DOI: 10.1137/0221023.

[232] L. Trevisan, "Extractors and pseudorandom generators," *J. ACM*, vol. 48, no. 4, 2001, pp. 860–879. DOI: 10.1145/502090.502099.

[233] L. Trevisan, "Pseudorandomness and combinatorial constructions," 2005. URL: https://web.archive.org/web/20150115081847/http://www.cs.berkeley.edu/~luca/pacc/.

[234] L. Trevisan and S. Vadhan, "Pseudorandomness and average-case complexity via uniform reductions," *Comput. Complexity*, vol. 16, no. 4, 2007, pp. 331–364. DOI: 10.1007/s00037-007-0233-x.

[235] L. Trevisan and T. Xue, "A derandomized switching lemma and an improved derandomization of AC0," in *Proc. 28th Annual IEEE Conference on Computational Complexity (CCC)*, pp. 242–247, 2013. DOI: 10.1109/CCC.2013.32.

[236] C. Umans, "Pseudo-random generators for all hardnesses," *J. of Computer and System Sciences*, vol. 67, no. 2, 2003, pp. 419–440. DOI: 10.1016/S0022-0000(03)00046-1.

[237] S. Vadhan and C. J. Zheng, "Characterizing pseudoentropy and simplifying pseudorandom generator constructions," in *Proc. 44th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 817–836, 2012. DOI: 10.1145/2213977.2214051.

[238] S. P. Vadhan, "Pseudorandomness," *Foundations and Trends in Theoretical Computer Science*, vol. 7, no. 1-3, 2012, pp. 1–336. DOI: 10.1561/0400000010.

[239] L. G. Valiant and V. V. Vazirani, "NP is as easy as detecting unique solutions," *Theoret. Comput. Sci.*, vol. 47, no. 1, 1986, pp. 85–93. DOI: 10.1016/0304-3975(86)90135-0.

[240] S. Vigna, "Further scramblings of Marsaglia's `xorshift` generators," *J. Comput. Appl. Math.*, vol. 315, 2017, pp. 175–181. DOI: 10.1016/j.cam.2016.11.006.

[241] E. Viola, "Pseudorandom bits for constant-depth circuits with few arbitrary symmetric gates," *SIAM J. Comput.*, vol. 36, no. 5, 2007, pp. 1387–1403. DOI: 10.1137/050640941.

[242] E. Viola, "The sum of D small-bias generators fools polynomials of degree D," *Comput. Complexity*, vol. 18, no. 2, 2009, pp. 209–217. DOI: 10.1007/s00037-009-0273-5.

[243] E. Viola, "Randomness buys depth for approximate counting," *Comput. Complexity*, vol. 23, no. 3, 2014, pp. 479–508. DOI: 10.1007/s00037-013-0076-6.

[244] E. Viola, "Special topics in complexity theory," 2017. URL: https://www.ccs.neu.edu/home/viola/classes/spepf17.html.

[245] E. Viola, "Fourier conjectures, correlation bounds, and majority," in *Proc. 48th International Colloquium on Automata, Languages and Programming (ICALP)*, 111:1–111:15, 2021. DOI: 10.4230/LIPIcs.ICALP.2021.111.

[246] E. Viola, *Correlation bounds against polynomials*, ECCC preprint TR22-142, 2022. URL: https://eccc.weizmann.ac.il/report/2022/142/.

[247] I. Wegener, *The complexity of Boolean functions*, ser. Wiley-Teubner Series in Computer Science. John Wiley & Sons, Inc., 1987, pp. xii+457. URL: https://dl.acm.org/doi/10.5555/35517.

[248] A. Wigderson, *Randomness and pseudorandomness*, IAS Institute Letter, 2009. URL: https://www.ias.edu/ideas/2009/wigderson-randomness-pseudorandomness.

[249] A. C. Yao, "Theory and applications of trapdoor functions," in *Proc. 23rd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 80–91, 1982. DOI: 10.1109/SFCS.1982.45.

[250] Y. Yu, D. Gu, X. Li, and J. Weng, "The randomized iterate, revisited—almost linear seed length PRGs from a broader class of one-way functions," in *Proc. 12th Theory of Cryptography Conference (TCC)*, pp. 7–35, 2015. DOI: 10.1007/978-3-662-46494-6_2.

[251] Y. Yu, X. Li, and J. Weng, "Pseudorandom generators from regular one-way functions: New constructions with improved parameters," *Theoret. Comput. Sci.*, vol. 569, 2015, pp. 58–69. DOI: 10.1016/j.tcs.2014.12.013.

[252] D. Zuckerman, "General weak random sources," in *Proc. 31st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 534–543, 1990. DOI: 10.1109/FSCS.1990.89574.

[253] D. Zuckerman, "Pseudorandomness and combinatorial constructions," 2001. URL: https://www.cs.utexas.edu/~diz/395T/01/.