Electronic Colloquium on Computational Complexity, Report No. 86 (2013)

Pseudorandomness for Regular Branching Programs via Fourier Analysis

Omer Reingold* Omer.Reingold@microsoft.com

Thomas Steinke† tsteinke@seas.harvard.edu

Salil Vadhan‡ salil@seas.harvard.edu

June 2013

Abstract

We present an explicit pseudorandom generator for oblivious, read-once, permutation branching programs of constant width that can read their input bits in any order. The seed length is $O(\log^2 n)$, where n is the length of the branching program. The previous best seed length known for this model was $n^{1/2+o(1)}$, which follows as a special case of a generator due to Impagliazzo, Meka, and Zuckerman (FOCS 2012) (which gives a seed length of $s^{1/2+o(1)}$ for arbitrary branching programs of size s). Our techniques also give seed length $n^{1/2+o(1)}$ for general oblivious, read-once branching programs of width $2^{n^{o(1)}}$, which is incomparable to the results of Impagliazzo et al.

Our pseudorandom generator is similar to the one used by Gopalan et al. (FOCS 2012) for read-once CNFs, but the analysis is quite different; ours is based on Fourier analysis of branching programs. In particular, we show that an oblivious, read-once, *regular* branching program of width w has Fourier mass at most $(2w)^{2k}$ at level k, independent of the length of the program.

*Microsoft Research Silicon Valley, 1065 La Avenida, Mountain View, CA.

†School of Engineering and Applied Sciences, Harvard University, 33 Oxford Street, Cambridge MA. Work done in part while at Stanford University. Supported by NSF grant CCF-1116616 and the Lord Rutherford Memorial Research Fellowship.

‡School of Engineering and Applied Sciences, Harvard University, 33 Oxford Street, Cambridge MA. Work done in part when on leave as a Visiting Researcher at Microsoft Research Silicon Valley and a Visiting Scholar at Stanford University. Supported in part by NSF grant CCF-1116616 and US-Israel BSF grant 2010196.

ISSN 1433-8092

1 Introduction

A major open problem in the theory of pseudorandomness is to construct an "optimal" pseudorandom generator for space-bounded computation. That is, we want an explicit pseudorandom generator that stretches a uniformly random seed of length $O(\log n)$ to n bits that cannot be distinguished from uniform by any $O(\log n)$-space algorithm (which receives the pseudorandom bits one at a time, in a streaming fashion, and may be nonuniform).

Such a generator would imply that every randomized algorithm can be derandomized with only a constant-factor increase in space (RL = L), and would also have a variety of other applications, such as in streaming algorithms [15], deterministic dimension reduction and SDP rounding [29], hashing [7], hardness amplification [12], almost k-wise independent permutations [16], and cryptographic pseudorandom generator constructions [11].

Unfortunately, for fooling general logspace algorithms, there has been essentially no improvement since the classic work of Nisan [21], which provided a pseudorandom generator of seed length $O(\log^2 n)$. Instead, a variety of works have improved the seed length for various restricted classes of logspace algorithms, such as algorithms that use $n^{o(1)}$ random bits [22, 23], combinatorial rectangles [9, 18, 2, 19] random walks on graphs [24, 25], branching programs of width 2 or 3 [27, 3, 32], and regular or permutation branching programs (of bounded width) [5, 6, 17, 8, 30].

The vast majority of these works are based on Nisan's generator or its variants by Impagliazzo, Nisan, and Wigderson [14] and Nisan and Zuckerman [22], and show how the analysis (and hence the final parameters) of these generators can be improved for logspace algorithms that satisfy the additional restrictions. All three of these generators are based on recursive use of the following principle: if we consider two consecutive time intervals I_1, I_2 in a space s computation and use some randomness r to generate the pseudorandom bits fed to the algorithm during interval I_1, then at the start of I_2, the algorithm will 'remember' at most s bits of information about r. So we can use a randomness extractor to extract roughly $|r| - s$ almost uniform bits from r (while investing only a small additional amount of randomness for the extraction). This paradigm seems unlikely to yield pseudorandom generators for general logspace computations that have a seed length of $\log^{1.99} n$ (see [6]).

Thus, there is a real need for a different approach to constructing pseudorandom generators for space-bounded computation. One new approach has been suggested in the recent work of Gopalan et al. [10], which constructed improved pseudorandom generators for read-once CNF formulas and combinatorial rectangles, and hitting set generators for width 3 branching programs. Their basic generator (e.g. for read-once CNF formulas) works as follows: Instead of considering a fixed partition of the bits into intervals, they pseudorandomly partition the bits into two groups, assign the bits in one group using a small-bias generator [20], and then recursively generate bits for the second group. While it would not work to assign *all* the bits using a single sample from a small-bias generator, it turns out that generating a pseudorandom partial assignment is a significantly easier task.

An added feature of the Gopalan et al. generator is that its pseudorandomness properties are independent of the order in which the output bits are read by a potential distinguisher. In contrast, Nisan's generator and its variants depend heavily on the ordering of bits (the intervals I_1 and I_2 above cannot be interleaved), and in fact it is known that a particular instantiation of Nisan's generator fails to be pseudorandom if the (space-bounded) distinguisher can read the bits in a different order [31, Corollary 3.18]. Recent works [4, 13] have constructed nontrivial pseudorandom generators for space-bounded algorithms that can read their bits in any order, but the seed length achieved is larger than $\sqrt{n}$.

In light of the above, a natural question is whether the approach of Gopalan et al. can be extended to a wider class of space-bounded algorithms. We make progress on this question by using the same approach to construct a pseudorandom generator with seed length $O(\log^2 n)$ for constant-width, read-once, oblivious permutation branching programs that can read their bits in any order. In analysing our generator, we

develop new Fourier-analytic tools for proving pseudorandomness against space-bounded algorithms.

1.1 Models of Space-Bounded Computation

A (layered) **branching program** B is a nonuniform model of space-bounded computation. The program maintains a **state** from the set $[w] = \{1, \ldots, w\}$ and, at each time step i, reads one bit of its input $x \in \{0,1\}^n$ and updates its state according to a transition function $B_i : \{0,1\} \times [w] \to [w]$. The parameter w is called the **width** of the program, and corresponds to a space bound of $\log w$ bits. We allow the transition function B_i to be different at each time step i. We consider several restricted forms of branching programs:

- **Read-once branching programs** read each input bit at most once.
- **Oblivious branching programs** choose which input bit to read depending only on the time step i, and not on the current state
- **Ordered branching programs** (a.k.a. streaming algorithms) always read input bit i in time step i (hence are necessarily both read-once and oblivious).

To derandomize randomized space-bounded computations (e.g. prove RL = L), it suffices to construct pseudorandom generators that fool ordered branching programs of polynomial width ($w = \text{poly}(n)$), and hence this is the model addressed by most previous constructions (including Nisan's generator). However, the more general models of oblivious and read-once branching programs are also natural to study, and, as discussed above, can spark the development of new techniques for reasoning about pseudorandomness.

As mentioned earlier, Nisan's pseudorandom generator [21] achieves $O(\log^2 n)$ seed length for *ordered* branching programs of polynomial width. It is known how to achieve $O(\log n)$ seed length for ordered branching programs width 2 [3], and for width 3, it is only known how to construct "hitting-set generators" (a weaker form of pseudorandom generators) with seed length $O(\log n)$ [32, 10]. (The seed length is $\tilde{O}(\log n)$ if we want the error of the hitting set generator to be subconstant.) For pseudorandom generators for width $w \geq 3$ and hitting-set generators for width $w \geq 4$, there is no known construction with seed length $o(\log^2 n)$.

The study of pseudorandomness against non-ordered branching programs started more recently. Tzur [31] showed that there are oblivious, read-once, constant-width branching programs that can distinguish the output of Nisan's generator from uniform. Bogdanov, Papakonstantinou, and Wan [4] exhibited a pseudorandom generator with seed length $(1 - \Omega(1)) \cdot n$ for oblivious read-once branching programs of width w for $w = 2^{\Omega(n)}$. Impagliazzo, Meka, and Zuckerman [13] gave a pseudorandom generator with seed length $s^{1/2+o(1)}$ for arbitrary branching programs of size s; note that $s = O(nw)$ for a read-once branching program of width w and length n.

We consider two further restrictions on branching programs:

- **Regular branching programs** are oblivious branching programs with the property that, if the distribution on states in any layer is uniformly random and the input bit read by the program at that layer is uniformly random, then the resulting distribution on states in the next layer is uniformly random. This is equivalent to requiring that the bipartite graph associated with each layer of the program, where we have edges from each state $u \in [w]$ in layer i to the possible next-states $u_0, u_1 \in [w]$ in layer $i+1$ (if the input bit is b, the state goes to u_b), is a regular graph.
- **Permutation branching programs** are a further restriction, where we require that for each setting of the input string, the mappings between layers are permutations. This is equivalent to saying that (regular) bipartite graphs corresponding to each layer are decomposed into two perfect matchings, one corresponding to each value of the current input bit being read.

The fact that pseudorandomness for permutation branching programs might be easier than for general branching programs was suggested by the proof that Undirected S-T Connectivity is in Logspace [24] and its follow-ups [25, 26]. Specifically, the latter works construct "pseudorandom walk generators" for "consistently labelled" graphs. Interpreted for permutation branching programs, these results ensure that if an ordered permutation branching program has the property that every layer has a nonnegligible amount of "mixing" — meaning that the distribution on states becomes closer to uniform, on a truly random input — then the overall program will also have mixing when run on the output of the pseudorandom generator (albeit at a slower rate). The generator has a seed length of $O(\log n)$ even for ordered permutation branching programs of width $\text{poly}(n)$. Reingold, Trevisan, and Vadhan [25] also show that if a generator with similar properties could be constructed for (ordered) regular branching programs of polynomial width, then this would suffice to prove RL = L. Thus, in the case of polynomial width, regularity is not a significant constraint.

Recently, there has been substantial progress on constructing pseudorandom generators for ordered regular and permutation branching programs of constant width. Braverman, Rao, Raz, and Yehudayoff [5] and Brody and Verbin [6] gave pseudorandom generators with seed length $\tilde{O}(\log n)$ for ordered regular branching programs of constant width. Koucký, Nimbhorkar and Pudlák [17] showed that the seed length could be further improved to $O(\log n)$ for ordered, permutation branching programs of constant width; see [8, 30] for simplifications and improvements.

All of these generators for ordered regular and permutation branching programs are based on refined analyses of the pseudorandom generator construction of Impagliazzo, Nisan, and Wigderson [14].

1.2 Our Results and Techniques

Our main result is a pseudorandom generator for read-once, oblivious, (unordered) permutation branching programs of constant width:

Theorem 1.1 (Main Result)**.** *For every constant w, there is an explicit pseudorandom generator $G : \{0,1\}^{O(\log^2 n)} \to \{0,1\}^n$ fooling oblivious, read-once (but unordered), permutation branching programs of width w and length n.*

To be precise, the seed length and space complexity of the pseudorandom generator is

$$O(w^2 \log(w) \log(n) \log(nw/\varepsilon) + w^4 \log^2(w/\varepsilon))$$

for oblivious, read-once, permutation branching programs of length n and width w, where ε is the error.

Previously, it was only known how to achieve a seed length of $n^{1/2+o(1)}$ for this model, as follows from the aforementioned results of Impagliazzo, Meka, and Zuckerman [13] (which actually holds for arbitrary branching programs).

Our techniques also achieve seed length $n^{1/2+o(1)}$ for arbitrary read-once, oblivious branching programs of width up to $2^{n^{o(1)}}$:

Theorem 1.2. *There is an explicit pseudorandom generator $G : \{0,1\}^{\tilde{O}(\sqrt{n} \log w)} \to \{0,1\}^n$ fooling oblivious, read-once (but unordered), permutation branching programs of width w and length n.*

This result is incomparable to that of Impagliazzo et al. [13]. Their seed length depends polynomially on the width w, so require width $w = n^{o(1)}$ to achieve seed length $n^{1/2+o(1)}$. On the other hand, our result is restricted to *read-once, oblivious* branching programs.

Our construction of the generator in Theorem 1.1 is essentially the same as the generator of Gopalan et al. [10] for read-once CNF formulas, but with a new analysis (and different setting of parameters) for read-once, oblivious, permutation branching programs. The generator works by selecting a subset $T \subset [n]$ of output

coordinates in a pseudorandom way, assigning the bits in T using another pseudorandom distribution X, and then recursively assigning the bits outside T. We generate T using an almost $O(\log n)$-wise independent distribution, including each coordinate $i \in T$ with a constant probability p_w depending only on the width w. We assign the bits in T using a small-bias distribution X on $\{0,1\}^n$ [20]; such a generator has the property that for every nonempty subset $S \subset [n]$, the parity $\oplus_{i \in S} X_i$ of bits in S has bias at most ε. Generating T requires $O(\log n)$ random bits, generating X requires $O(\log n)$ bits (even for $\varepsilon = 1/\mathrm{poly}(n)$), and we need $O(\log n)$ levels of recursion to assign all the bits. This gives us our $O(\log^2 n)$ seed length.

Let $B : \{0,1\}^n \to \{0,1\}$ be a function computed by an oblivious, read-once, permutation branching program of width w. Following [10], to show that our pseudorandom generator fools B, it suffices to show that the partial assignment generated in a single level of recursion approximately preserves the acceptance probability of B (on average). To make this precise, we need a bit of notation. For a set $t \subset [n]$, a string $x \in \{0,1\}^n$, and $y \in \{0,1\}^{n-|t|}$, define $\mathrm{Select}(t,x,y) \in \{0,1\}^n$ as follows:

$$\mathrm{Select}(t,x,y)_i = \begin{cases} x_i & \text{if } i \in t \\ y_{|\{j \le i : j \notin t\}|} & \text{if } i \notin t \end{cases}$$

Once we choose a set $t \leftarrow T$ and an assignment $x \leftarrow X$ to the variables in t, the residual acceptance probability of B is $\mathop{\mathbb{P}}_{U}[B(\mathrm{Select}(t,x,U)) = 1]$, where U is the uniform distribution on $\{0,1\}^n$. So, the average acceptance probability over $t \leftarrow T$ and $x \leftarrow X$ is $\mathop{\mathbb{P}}_{T,X,U}[B(\mathrm{Select}(T,X,U)) = 1]$. We would like this to be close to the acceptance probability under uniformly random bits, namely $\mathop{\mathbb{P}}_{U}[B(U) = 1] = \mathop{\mathbb{P}}_{T,U',U}[B(\mathrm{Select}(T,U',U) = 1]$. That is, we would like our small-bias distribution X to fool the function $B'(x) := \mathop{\mathbb{E}}_{T,U}[B(\mathrm{Select}(T,x,U))]$. The key insight in [10] is that B' can be a significantly easier function to fool than B, and even than fixed restrictions of B (like $B(\mathrm{Select}(t,\cdot,y))$ for fixed t and y). We show that the same phenomenon holds for oblivious, read-once, *regular* branching programs. (The reason that the analysis of our overall pseudorandom generator applies only for *permutation* branching programs is that regularity is not preserved under restriction (as needed for the recursion), whereas the permutation property is.)

To show that a small-bias space fools $B'(x)$, it suffices to show that the **Fourier mass** of B', namely $\sum_{s \in \{0,1\}^n, s \neq 0} |\widehat{B'}[s]|$, is bounded by $\mathrm{poly}(n)$. (Here $\widehat{B'}[s] = \mathop{\mathbb{E}}_{U}\left[B'[U] \cdot (-1)^{s \cdot U}\right]$ is the standard Fourier transform over $\mathbb{Z}_2^n$. So $\widehat{B'}[s]$ measures the correlation of B' with the parity function defined by s.) We show that this is indeed the case (for most choices of the set $t \leftarrow T$):

Theorem 1.3 (Main Lemma)**.** *For every constant w, there are constants $p_w > 0$ and $d_w \in \mathbb{N}$ such that the following holds. Let $B : \{0,1\}^n \to \{0,1\}$ be computed by an oblivious, read-once, regular branching program of width w and length $n \ge d_w$. Let $T \subset [n]$ be a randomly chosen set so that every coordinate $i \in [n]$ is placed in T with probability p_w and these choices are n^{-d_w}-almost $(d_w \log n)$-wise independent. Then with high probability over $t \leftarrow T$ $B'(x) = \mathop{\mathbb{E}}_{U}[B(\mathrm{Select}(t,x,U))]$ has Fourier mass at most n^{d_w}.*

As a warm-up, we begin by analysing the Fourier mass in the case the set T is chosen completely at random, with every coordinate included independently with probability p_w. In this case, it is more convenient to average over T and work with $B'(x) = \mathop{\mathbb{E}}_{T,U}[B(\mathrm{Select}(T,x,U))]$. Then it turns out that $\widehat{B'}[s] = p_w^{|s|} \cdot \widehat{B}[s]$, where $|s|$ denotes the Hamming weight of the vector s. Thus, it suffices to analyse the original program B and show that for each $k \in \{1, \cdots, n\}$, the Fourier mass of B restricted to s of weight k is at most c_w^k, where c_w is a constant depending only on w (not on n). We prove that this is indeed the case for regular branching programs:

Theorem 1.4. *Let $B : \{0,1\}^n \to \{0,1\}$ be a function computed by an oblivious, read-once, regular branching program of width w. Then for every $k \in \{1, \ldots, n\}$, we have*

$$\sum_{s \in \{0,1\}^n : |s| = k} |\widehat{B}[s]| \le (2w^2)^k.$$

Our proof of Theorem 1.4 relies on the main lemma of Braverman et al. [5], which intuitively says that in a bounded-width, read-once, oblivious, regular branching program, only a constant number of bits have a significant effect on the acceptance probability. More formally, if we sum, for every time step i and all possible states v at time i, the absolute difference between the acceptance probability after reading a 0 versus reading a 1 from state v, the total will be bounded by $\mathrm{poly}(w)$ (independent of n). This directly implies a bound of $\mathrm{poly}(w)$ on the Fourier mass of B at the first level: the correlation of B with a parity of weight 1 is bounded by the effect of a single bit on the output of B. We then bound the correlation of B with a parity of weight k by the correlation of a *prefix* of B with a parity of weight $k-1$ times the effect of the remaining bit on B. Thus we inductively obtain the bound on the Fourier mass of B at level k.

Our proof of Theorem 1.3 for the case of a pseudorandom restriction T uses the fact that we can decompose the high-order Fourier coefficients of an oblivious, read-once branching program B' into products of low-order Fourier coefficients of "subprograms" (intervals of consecutive layers) of B'. Using an almost $O(\log n)$-wise independent choice of T enables us to control the Fourier mass at level $O(\log n)$ for all subprograms of B', which suffices to control the total Fourier mass of B'.

1.3 Organization

In Section 2 we introduce the definitions and tools we use in our proof. In Section 2.1 we formally define branching programs and explain our view of them as matrix-valued functions. In Sections 2.3 and 2.5 we define the matrix-valued Fourier transform and explain how we use it.

Our results use Fourier analysis of regular branching programs to analyse pseudorandom generators. In Section 3, we give a bound on the low-order Fourier coefficients of a read-once, oblivious, regular branching program (Theorem 1.4/3.2) using the main lemma of Braverman et al. [5]. This yields a result about random restrictions, which we define and discuss in Section 4. We extend the results about random restrictions to pseudorandom restrictions in Section 5 and prove our main lemma (Theorem 1.3/5.2). Finally, in Section 6 we construct and analyse our pseudorandom generator, which proves the main result (Theorem 1.1/6.1).

In Section 7 we show how to extend our techniques to general read-once, oblivious branching programs (Theorem 1.2/7.1). We conclude in Section 8 by discussing directions for further work.

2 Preliminaries

2.1 Branching Programs

We define a length-n, width-w **program** to be a function $B : \{0,1\}^n \times [w] \to [w]$, which takes a start state $u \in [w]$ and an input string $x \in \{0,1\}^n$ and outputs a final state $B[x](u)$.

Often we think of B as having a fixed **start state** u_0 and a set of **accept states** $S \subset [w]$. Then B **accepts** $x \in \{0,1\}^n$ if $B[x](u_0) \in S$. We say that B **computes the function** $f : \{0,1\}^n \to \{0,1\}$ if $f(x) = 1$ if and only if $B[x](u_0) \in S$.

In our applications, the input x is randomly (or pseudorandomly) chosen, in which case a program can be viewed as a Markov chain randomly taking initial states to final states. For each $x \in \{0,1\}^n$, we let $B[x] \in \{0,1\}^{w \times w}$ be a matrix defined by

$$B[x](u,v) = 1 \iff B[x](u) = v.$$

For a random variable X on $\{0,1\}^n$, we have $\mathop{\mathbb{E}}_X [B[X]] \in [0,1]^{w \times w}$, where $\mathop{\mathbb{E}}_R [f(R)]$ is the **expectation** of a function f with respect to a random variable R. Then the entry in the u^{th} row and v^{th} column $\mathop{\mathbb{E}}_X [B[X]] (u,v)$

is the probability that B takes the initial state u to the final state v when given a random input from the distribution X—that is,

$$\mathop{\mathbb{E}}_{X}[B[X]](u,v) = \mathop{\mathbb{P}}_{X}[B[X](u) = v],$$

where $\mathop{\mathbb{P}}_{R}[e(R)]$ is the **probability** of an event e with respect to the random variable R.

A branching program reads one bit of the input at a time (rather than reading x all at once) maintaining only a state in $[w] = \{1, 2, \cdots, w\}$ at each step. We capture this restriction by demanding that the program be composed of several smaller programs, as follows.

Let B and B' be width-w programs of length n and n' respectively. We define the **concatenation** $B \circ B' : \{0,1\}^{n+n'} \times [w] \to [w]$ of B and B' by

$$(B \circ B')[x \circ x'](u) := B'[x'](B[x](u)),$$

which is a width-w, length-$(n + n')$ program. That is, we run B and B' on separate inputs, but the final state of B becomes the start state of B'. Concatenation corresponds to matrix multiplication—that is, $(B \circ B')[x \circ x'] = B[x] \cdot B'[x']$, where the two programs are concatenated on the left hand side and the two matrices are multiplied on the right hand side.

A length-n, width-w, **ordered branching program** is a program B that can be written $B = B_1 \circ B_2 \circ \cdots \circ B_n$, where each B_i is a length-1 width-w program. We refer to B_i as the i^{th} **layer** of B. We denote the **subprogram** of B from layer i to layer j by $B_{i \cdots j} := B_i \circ B_{i+1} \circ \cdots \circ B_j$.

General read-once, oblivious branching programs (a.k.a. unordered branching programs) can be reduced to the ordered case by a permutation of the input bits. Formally, a **read-once, oblivious branching program** B is an ordered branching program B' composed with a permutation π. That is, $B[x] = B'[\pi(x)]$, where the i^{th} bit of $\pi(x)$ is the $\pi(i)^{\text{th}}$ bit of x

For a program B and an arbitrary distribution X, the matrix $\mathop{\mathbb{E}}_{X}[B[X]]$ is **stochastic**—that is, $\sum_v \mathop{\mathbb{E}}_{X}[B[X]](u,v) = 1$ for all u and $\mathop{\mathbb{E}}_{X}[B[X]](u,v) \geq 0$ for all u and v. A program B is called a **regular program** if the matrix $\mathop{\mathbb{E}}_{U}[B[U]]$ is **doubly stochastic**—that is, both $\mathop{\mathbb{E}}_{U}[B[U]]$ and its transpose $\mathop{\mathbb{E}}_{U}[B[U]]^*$ are stochastic. A program B is called a **permutation program** if $B[x]$ is a permutation matrix for every x or, equivalently, $B[x]$ is doubly stochastic. Note that a permutation program is necessarily a regular program and, if both B and B' are regular or permutation programs, then so is their concatenation.

A regular program B has the property that the uniform distribution is a stationary distribution of the Markov chain $\mathop{\mathbb{E}}_{U}[B[U]]$, whereas, if B is a permutation program, the uniform distribution is stationary for $\mathop{\mathbb{E}}_{X}[B[X]]$ for *any* distribution X.

A **regular branching program** is a branching program where each layer B_i is a regular program and likewise for a **permutation branching program**.

2.2 Norms

We are interested in constructing a random variable X (the output of the pseudorandom generator) such that $\mathop{\mathbb{E}}_{X}[B[X]] \approx \mathop{\mathbb{E}}_{U}[B[U]]$, where U is uniform on $\{0,1\}^n$. Throughout we use U to denote the **uniform distribution**. The error of the pseudorandom generator will be measured by the norm of the matrix $\mathop{\mathbb{E}}_{X}[B[X]] - \mathop{\mathbb{E}}_{U}[B[U]]$.

For a matrix $A \in \mathbb{R}^{w \times w}$, define the ρ **operator norm** of A by

$$||A||_\rho = \max_x \frac{||xA||_\rho}{||x||_\rho},$$

where ρ specifies a vector norm (usually 1, 2, or ∞ norm). Define the **Frobenius norm** of $A \in \mathbb{R}^{w \times w}$ by

$$||A||_{\text{Fr}}^2 = \sum_{u,v} A(u,v)^2 = \text{trace}(A^* A) = \sum_{\lambda} |\lambda|^2,$$

where A^* is the (conjugate) transpose of A and the last sum is over the singular values λ of A. Note that $||A||_2 \leq ||A||_{\text{Fr}}$ for all A.

We almost exclusively use the Euclidean norm ($||x||_2 = \sqrt{\sum_i x(i)^2}$) and the corresponding spectral norm ($||A||_2 = \max_\lambda |\lambda|$). This is not crucial; our results would work with any reasonable norm.

2.3 Fourier Analysis

Let $B : \{0,1\}^n \to \mathbb{R}^{w \times w}$ be a matrix-valued function (such as given by a length-n, width-w branching program). Then we define the **Fourier transform** of B as a matrix-valued function $\widehat{B} : \{0,1\}^n \to \mathbb{R}^{w \times w}$ given by

$$\widehat{B}[s] := \mathop{\mathbb{E}}_{U} \left[B[U] \chi_s(U) \right],$$

where $s \in \{0,1\}^n$ (or, equivalently, $s \subset [n]$) and

$$\chi_s(x) = (-1)^{\sum_i x(i) \cdot s(i)} = \prod_{i \in s} (-1)^{x(i)}.$$

We refer to $\widehat{B}[s]$ as the s^{th} **Fourier coefficient** of B. The **order** of a Fourier coefficient $\widehat{B}[s]$ is $|s|$—the **Hamming weight** of s, which is the size of the set s or the number of 1s in the string s. Note that this is equivalent to taking the real-valued Fourier transform of each of the w^2 entries of B separately, but we will see below that this matrix-valued Fourier transform is nicely compatible with matrix algebra.

For a random variable X over $\{0,1\}^n$ we define its s^{th} Fourier coefficient as

$$\widehat{X}(s) := \mathop{\mathbb{E}}_{X} \left[\chi_s(X) \right],$$

which, up to scaling, is the same as taking the real-valued Fourier transform of the probability mass function of X. We have the following useful properties.

Lemma 2.1. *Let $A, B : \{0,1\}^n \to \mathbb{R}^{w \times w}$ be matrix valued functions. Let X, Y, and U be independent random variables over $\{0,1\}^n$, where U is uniform. Let $s, t \in \{0,1\}^n$. Then we have the following.*

- *Decomposition: If $C[x \circ y] = A[x] \cdot B[y]$ for all $x, y \in \{0,1\}^n$, then $\widehat{C}[s \circ t] = \widehat{A}[s] \cdot \widehat{B}[t]$.*
- *Expectation: $\mathop{\mathbb{E}}_{X} [B[X]] = \sum_s \widehat{B}[s] \widehat{X}(s)$.*
- *Fourier Inversion for Matrices: $B[x] = \sum_s \widehat{B}[s] \chi_s(x)$.*
- *Fourier Inversion for Distributions: $\mathop{\mathbb{P}}_{X} [X = x] = \mathop{\mathbb{E}}_{U} \left[\widehat{X}(U) \chi_U(x) \right]$.*
- *Convolution for Distributions: If $Z = X \oplus Y$, then $\widehat{Z}(s) = \widehat{X}(s) \cdot \widehat{Y}(s)$.*
- *Parseval's Identity: $\sum_{s \in \{0,1\}^n} \left|\left| \widehat{B}[s] \right|\right|_{Fr}^2 = \mathop{\mathbb{E}}_{U} \left[||B[U]||_{Fr}^2 \right]$.*
- *Convolution for Matrices: If, for all $x \in \{0,1\}^n$, $C[x] = \mathop{\mathbb{E}}_{U} [A[U] \cdot B[U \oplus x]]$, then $\widehat{C}[s] = \widehat{A}[s] \cdot \widehat{B}[s]$.*

The Decomposition property is what makes the matrix-valued Fourier transform more convenient than separately taking the Fourier transform of the matrix entries as done in [4]. If B is a length-n width-w branching program, then, for all $s \in \{0,1\}^n$,

$$\widehat{B}[s] = \widehat{B}_1[s_1] \cdot \widehat{B}_2[s_2] \cdot \cdots \cdot \widehat{B}_n[s_n].$$

2.4 Small-Bias Distributions

The **bias** of a random variable X over $\{0,1\}^n$ is defined as

$$\text{bias}(X) := \max_{s \neq 0} |\widehat{X}(s)|.$$

A distribution is **ε-biased** if it has bias at most ε. Note that a distribution has bias 0 if and only if it is uniform. Thus a distribution with small bias is an approximation to the uniform distribution. We can sample an ε-biased distribution X on $\{0,1\}^n$ with seed length $O(\log(n/\varepsilon))$ and using space $O(\log(n/\varepsilon))$ [20, 1].

Small-bias distributions are useful pseudorandom generators: A ε-biased random variable X is indistinguishable from uniform by any linear function (a parity of a subset of the bits of X). That is, for any $s \subset [n]$, we have $\left| \mathop{\mathbb{E}}_{X} \left[\bigoplus_{i \in s} X_i \right] - 1/2 \right| \leq 2\varepsilon$. Small bias distributions are known to be good pseudorandom generators for width-2 branching programs [3], but not width-3. For example, the uniform distribution over $\{x \in \{0,1\}^n : |x| \mod 3 = 0\}$ has bias $2^{-\Omega(n)}$, but does not fool width-3, ordered, permutation branching programs.

2.5 Fourier Mass

We analyse small bias distributions as pseudorandom generators for branching programs using Fourier analysis. Intuitively, the Fourier transform of a branching program expresses that program as a linear combination of linear functions (parities), which can then be fooled using a small-bias space.

Define the **Fourier mass** of a matrix-valued function B to be

$$L_\rho(B) := \sum_{s \neq 0} \left|\left| \widehat{B}[s] \right|\right|_\rho.$$

Also, define the **Fourier mass of B at level k** as

$$L_\rho^k(B) := \sum_{s \in \{0,1\}^n : |s| = k} \left|\left| \widehat{B}[s] \right|\right|_\rho.$$

Note that $L_\rho(B) = \sum_{k \geq 1} L_\rho^k(B)$.

The Fourier mass is unaffected by order:

Lemma 2.2. *Let $B, B' : \{0,1\}^n \to \mathbb{R}^{w \times w}$ be matrix-valued functions satisfying $B[x] = B'[\pi(x)]$, where $\pi : [n] \to [n]$ is a permutation. Then, for all $s \in \{0,1\}^n$, $\widehat{B}[s] = \widehat{B'}[\pi(s)]$. In particular, $L_\rho(B) = L_\rho(B')$ and $L_\rho^k(B) = L_\rho^k(B')$ for all k and ρ.*

Lemma 2.2 implies that the Fourier mass of any read-once, oblivious branching program is equal to the Fourier mass of the corresponding ordered branching program.

If $L_\rho(B)$ is small, then B is fooled by a small-bias distribution:

Lemma 2.3. *Let B be a length-n, width-w branching program. Let X be a ε-biased random variable on $\{0,1\}^n$. For any matrix norm $||\cdot||_\rho$, we have*

$$\left|\left|\mathop{\mathbb{E}}_{X}[B[X]] - \mathop{\mathbb{E}}_{U}[B[U]]\right|\right|_\rho = \left|\left|\sum_{s\neq 0} \widehat{B}[s]\widehat{X}(s)\right|\right|_\rho \leq L_\rho(B)\varepsilon.$$

In the worst case $L_2(B) = 2^{\Theta(n)}$, even for a length-n width-3 permutation branching program B. For example, the program $B_{\mathrm{mod}\ 3}$ that computes the Hamming weight of its input modulo 3 has exponential Fourier mass.

We show that, using 'restrictions', we can ensure that $L_\rho(B)$ is small.

3 Fourier Analysis of Regular Branching Programs

We use a result by Braverman et al. [5]. The following is a Fourier-analytic reformulation of their result.

Lemma 3.1 ([5, Lemma 4])**.** *Let B be a length-n, width-w, ordered, regular branching program. Then*

$$\sum_{1\leq i\leq n} \left|\left|\widehat{B_{i\cdots n}}[1\circ 0^{n-i}]\right|\right|_2 \leq 2w^2.$$

Braverman et al. instead consider the sum, over all $i \in [n]$ and all states $u \in [w]$ at layer i, of the difference in acceptance probabilities if we run the program starting at v with a 0 followed by random bits versus a 1 followed by random bits. They refer to this quantity as the weight of B. Their result can be expressed in Fourier-analytic terms as follows:

$$\sum_{1\leq i\leq n} \left|\left|\widehat{B_{i\cdots n}}[1\circ 0^{n-i}]q\right|\right|_1 \leq 2(w-1)$$

for any $q \in \{0,1\}^w$ with $|q| = 1$. (The vector q can be used to specify the accept states of B, and the v^{th} row of $\widehat{B_{i\cdots n}}[1\circ 0^{n-i}]q$ is precisely the difference in acceptance probabilities mentioned above.)

By summing over all possible q, we obtain Lemma 3.1. For completeness, we include a proof of Lemma 3.1 in Appendix A.

Lemma 3.1 is similar (but not identical) to a bound on the first-order Fourier coefficients of a regular branching program: The term $\widehat{B_{i\cdots n}}[1\circ 0^{n-i}]$ measures the effect of the i^{th} bit on the output of B when we start the program at layer i, whereas the i^{th} first-order Fourier coefficient $\widehat{B}[0^{i-1}\circ 1\circ 0^{n-i}]$ measures the effect of the i^{th} bit when we start at the first layer and run the first $i-1$ layers with random bits. This difference allows us to use Lemma 3.1 to obtain a bound on all low-order Fourier coefficients:

Theorem 3.2. *Let B be a length-n, width-w, read-once, oblivious, regular branching program. Then*

$$L_2^k(B) := \sum_{s\in\{0,1\}^n : |s|=k} \left|\left|\widehat{B}[s]\right|\right|_2 \leq (2w^2)^k.$$

The key point is that the bound does not depend on n, even though we are summing $\binom{n}{k}$ terms.

Proof. By Lemma 2.2, we may assume that B is ordered. We perform an induction on k. If $k = 0$, then there is only one Fourier coefficient to bound—namely, $\widehat{B}[0^n] = \mathop{\mathbb{E}}_{U}[B[U]]$. Since $\mathop{\mathbb{E}}_{U}[B[U]]$ is doubly stochastic, the

base case follows. Now suppose the bound holds for k and consider $k+1$. We split the Fourier coefficients based on where the last 1 is:

$$\begin{aligned}
&\sum_{s\in\{0,1\}^n:|s|=k+1} \left|\left|\widehat{B}[s]\right|\right|_2 \\
&= \sum_{1\le i\le n} \sum_{s\in\{0,1\}^{i-1}:|s|=k} \left|\left|\widehat{B}[s\circ 1\circ 0^{n-i}]\right|\right|_2 \\
&= \sum_{1\le i\le n} \sum_{s\in\{0,1\}^{i-1}:|s|=k} \left|\left|\widehat{B_{1\cdots i-1}}[s]\cdot\widehat{B_{i\cdots n}}[1\circ 0^{n-i}]\right|\right|_2 \quad \text{(by Lemma 2.1 (Decomposition))} \\
&\le \sum_{1\le i\le n} \sum_{s\in\{0,1\}^{i-1}:|s|=k} \left|\left|\widehat{B_{1\cdots i-1}}[s]\right|\right|_2 \cdot \left|\left|\widehat{B_{i\cdots n}}[1\circ 0^{n-i}]\right|\right|_2 \\
&\le \sum_{1\le i\le n} (2w^2)^k \cdot \left|\left|\widehat{B_{i\cdots n}}[1\circ 0^{n-i}]\right|\right|_2 \quad \text{(by the induction hypothesis)} \\
&\le (2w^2)^k\cdot 2w^2 \quad \text{(by Lemma 3.1)} \\
&= (2w^2)^{k+1},
\end{aligned}$$

as required. □

4 Random Restrictions

Our results involve restricting branching programs. However, our use of restrictions is different from elsewhere in the literature. Here, as in [10], we use (pseudorandom) restrictions in the usual way, but we *analyse* them by averaging over the *unrestricted* bits. Formally, we define a restriction as follows.

Definition 4.1. *For $t\in\{0,1\}^n$ and a length-n branching program B, let $B|_t$ be the **restriction** of B to t—that is, $B|_t:\{0,1\}^n\to\mathbb{R}^{w\times w}$ is a matrix-valued function given by $B|_t[x]:=\mathop{\mathbb{E}}_U[B[\mathrm{Select}(t,x,U)]]$, where U is uniform on $\{0,1\}^n$.*

Here Select takes a set $t\subset[n]$, a string $x\in\{0,1\}^n$, and a string y of length at least $n-|t|$ and produces a string of length n given by

$$\mathrm{Select}(t,x,y)(i)=\left\{\begin{array}{ll} x(i) & i\in t \\ y(|[i]\backslash t|) & i\in[n]\backslash t\end{array}\right\}.$$

Intuitively, $\mathrm{Select}(t,x,y)$ 'stretches' y by 'skipping' the bits in t and using bits from x instead. For example, $\mathrm{Select}(0101000,1111111,00001)=0101001$.

The most important aspect of restrictions is how they relate to the Fourier transform: For all B, s, and t, we have $\widehat{B|_t}[s]=\widehat{B}[s]$ if $s\subset t$ and $\widehat{B|_t}[s]=0$ otherwise. The restriction t 'kills' all the Fourier coefficients that are not contained in it. This means that a restriction significantly reduces the Fourier mass:

Lemma 4.2. *Let B be a length-n, width-w program. Let T be n independent random bits each with expectation p. Then*

$$\mathop{\mathbb{E}}_T[L_2(B|_T)]=\sum_{s\neq 0}p^{|s|}\left|\left|\widehat{B}[s]\right|\right|_2.$$

Proof.

$$\mathop{\mathbb{E}}_T[L_2(B|_T)]=\mathop{\mathbb{E}}_T\left[\sum_{s\neq 0:s\subset T}\left|\left|\widehat{B}[s]\right|\right|_2\right]=\sum_{s\neq 0}\mathop{\mathbb{P}}_T[s\subset T]\left|\left|\widehat{B}[s]\right|\right|_2=\sum_{s\neq 0}p^{|s|}\left|\left|\widehat{B}[s]\right|\right|_2.$$

□

We will overload notation as follows. Let $B = B' \circ B''$ be a branching program, where B' has length n' and B'' has length n'' and B has length $n = n' + n''$. For $t \in \{0,1\}^n$, we define $B'|_t = B'|_{t'}$ and $B''|_t = B'|_{t''}$ where $t' \in \{0,1\}^{n'}$, $t'' \in \{0,1\}^{n''}$ and $t = t' \circ t''$. Then $B|_t = B'|_{t'} \circ B''|_{t''} = B'|_t \circ B''|_t$.

Now we use Theorem 3.2 to prove a result about random restrictions:

Proposition 4.3. *Let B be a length-n, width-w, read-once, oblivious, regular branching program. Let T be n independent random bits each with expectation $p < 1/2w^2$. Then*

$$\mathop{\mathbb{E}}_{T}\left[L_2(B|_T)\right] \leq \frac{2w^2 \cdot p}{1 - 2w^2 \cdot p}.$$

In particular, if $p \leq 1/4w^2$, then $\mathop{\mathbb{E}}_{T}\left[L_2(B|_T)\right] \leq 1$.

Proof. By Lemma 4.2, we have,

$$\begin{aligned}
\mathop{\mathbb{E}}_{T}\left[L_2(B|_T)\right] &= \sum_{s \neq 0} p^{|s|} \cdot \left|\left|\widehat{B}[s]\right|\right|_2 \\
&= \sum_{1 \leq k \leq n} p^k \cdot \sum_{s:|s|=k} \left|\left|\widehat{B}[s]\right|\right|_2 \\
&\leq \sum_{1 \leq k \leq n} p^k \cdot (2w^2)^k \qquad \text{(by Theorem 3.2)} \\
&\leq \sum_{k \geq 1} (2w^2 \cdot p)^k \\
&= \frac{2w^2 \cdot p}{1 - 2w^2 \cdot p}.
\end{aligned}$$

□

Relation to Coin Theorem The Coin Theorem of Brody and Verbin [6] shows that general (non-regular) oblivious, read-once branching programs of width w cannot distinguish n independent and unbiased coin flips from ones with bias $1/(\log n)^{\Theta(w)}$, and they show that this bound is the best possible. Braverman et al. [5] show that regular, oblivious, read-once branching programs of width w cannot distinguish coins with bias $\Theta(1/w)$ from unbiased ones. (They state this result in terms of α-biased spaces.)

If Z is n independent coin flips with bias p and B is a branching program, then

$$\left|\left|\mathop{\mathbb{E}}_{Z}\left[B[Z]\right] - \mathop{\mathbb{E}}_{U}\left[B[U]\right]\right|\right| = \left|\left|\sum_{s \neq 0} \widehat{B}[s]\widehat{Z}(s)\right|\right| = \left|\left|\sum_{s \neq 0} p^{|s|}\widehat{B}[s]\right|\right|.$$

Thus Proposition 4.3 implies a Coin Theorem showing that read-once, oblivious, regular branching programs cannot distinguish coins with bias p for some $p = \Theta(1/w^2)$ from unbiased ones. This Coin Theorem is weaker than the Braverman et al. result, which gives $p = \Theta(1/w)$. However, Proposition 4.3 gives more than a Coin Theorem, as the sum is taken outside the norm—that is, we bound

$$\sum_{s \neq 0} p^{|s|} \left|\left|\widehat{B}[s]\right|\right|_2.$$

This distinction is important for our purposes, as it will allow us to reason about small-bias distributions and restrictions together.

5 Pseudorandom Restrictions

To analyse our generator, we need a pseudorandom version of Proposition 4.3. That is, we need to prove that, for a *pseudorandom* T (generated using few random bits), $L_2(B|_T)$ is small. We will generate T using an almost $O(\log n)$-wise independent distribution:

Definition 5.1. *A random variable X on Ω^n is δ-**almost** k-**wise independent** if, for any $I = \{i_1, i_2, \cdots, i_k\} \subset [n]$ with $|I| = k$, the coordinates $(X_{i_1}, X_{i_2}, \cdots, X_{i_k}) \in \Omega^k$ are δ statistically close to being independent—that is, for all $T \subset \Omega^k$,*

$$\left| \sum_{x \in T} \left(\mathop{\mathbb{P}}_{X} [(X_{i_1}, X_{i_2}, \cdots, X_{i_k}) = x] - \prod_{l \in [k]} \mathop{\mathbb{P}}_{X} [X_{i_l} = x_l] \right) \right| \leq \delta.$$

*We say that X is k-**wise independent** if it is 0-almost k-wise independent.*

We can sample a random variable X on $\{0,1\}^n$ that is δ-almost k-wise independent such that each bit has expectation $p = 2^{-d}$ using $O(kd + \log(1/\delta) + d\log(nd))$ random bits. See Lemma B.2 for more details.

Theorem 5.2 (Main Lemma)**.** *Let B be a length-n, width-w, read-once, oblivious, regular branching program. Let T be a random variable over $\{0,1\}^n$ where each bit has expectation p and the bits are δ-almost $2k$-wise independent. Suppose $p \leq (2w)^{-2}$ and $\delta \leq (2w)^{-4k}$. Then*

$$\mathop{\mathbb{P}}_{T} \left[L_2(B|_T) \leq (2w^2)^k \right] \geq 1 - n^4 \cdot \frac{2}{2^k}.$$

In particular, we show that, for $w = O(1)$, $k = O(\log n)$, and $\delta = 1/\mathrm{poly}(n)$, we have $L_2(B|_T) \leq \mathrm{poly}(n)$ with probability $1 - 1/\mathrm{poly}(n)$.

First we show that the Fourier mass at level $O(\log n)$ is bounded by $1/n$ with high probability. This also applies to all subprograms—that is,

$$\mathop{\mathbb{P}}_{T} \left[\forall i, j \quad L_2^k(B_{i \cdots j}|_T) \leq 1/n \right] \geq 1 - 1/\mathrm{poly}(n).$$

Lemma 5.3. *Let B be a length-n, width-w, ordered, regular branching program. Let T be a random variable over $\{0,1\}^n$ where each bit has expectation p and the bits are δ-almost k-wise independent. If $p \leq (2w)^{-2}$ and $\delta \leq (2w)^{-2k}$, then, for all $\beta > 0$,*

$$\mathop{\mathbb{P}}_{T} \left[\forall 1 \leq i \leq j \leq n \quad L_2^k(B_{i \cdots j}|_T) \leq \beta \right] \geq 1 - n^2 \frac{2}{2^k \beta}.$$

Proof. By Theorem 3.2, for all i and j,

$$\mathop{\mathbb{E}}_{T} \left[L_2^k(B_{i \cdots j}|_T) \right] = \sum_{s \subset \{i \cdots j\} : |s| = k} \mathop{\mathbb{P}}_{T} [s \subset T] \left\| \widehat{B_{i \cdots j}}[s] \right\|_2 \leq (2w^2)^k (p^k + \delta) \leq \frac{2}{2^k}.$$

The result now follows from Markov's inequality and a union bound. □

Now we use Lemma 5.3 to bound the Fourier mass at higher levels. We decompose high-order ($k' \geq 2k$) Fourier coefficients into low-order ($k \leq k' < 2k$) ones:

Lemma 5.4. *Let B be a length-n, ordered branching program and $t \in \{0,1\}^n$. Suppose that, for all i, j, and k' with $1 \leq i \leq j \leq n$ and $k \leq k' < 2k$, $L_2^{k'}(B_{i \cdots j}|_t) \leq 1/n$. Then, for all $k'' \geq k$ and all i and j, $L_2^{k''}(B_{i \cdots j}|_t) \leq 1/n$.*

Proof. Suppose otherwise and let k'' be the smallest $k'' \geq k$ such that $L_2^{k''}(B_{i\cdots j}|_t) > 1/n$ for some i and j. Clearly $k'' \geq 2k$. So, by minimality, the result holds for $k'' - k$. Fix i and j. Now

$$\begin{aligned}
L_2^{k''}(B_{i\cdots j}|_t) &= \sum_{s\in\{0,1\}^{j-i+1}:|s|=k''} \left|\left|\widehat{B_{i\cdots j}}[s]\right|\right|_2 \\
&\leq \sum_{l\in\{i\cdots j\}} \sum_{s\in\{0,1\}^{l-i+1}:|s|=k} \sum_{s'\in\{0,1\}^{j-l}:|s'|=k''-k} \left|\left|\widehat{B_{i\cdots l}}[s]\cdot\widehat{B_{l+1\cdots j}}[s']\right|\right|_2 \\
&\leq \sum_{l\in\{i\cdots j\}} \left(\sum_{s\in\{0,1\}^{l-i+1}:|s|=k} \left|\left|\widehat{B_{i\cdots l}}[s]\right|\right|_2\right)\left(\sum_{s'\in\{0,1\}^{j-l}:|s'|=k''-k} \left|\left|\widehat{B_{l+1\cdots j}}[s']\right|\right|_2\right) \\
&\leq \sum_{l\in\{i\cdots j\}} \frac{1}{n^2} \\
&\leq \frac{1}{n}.
\end{aligned}$$

Since i and j were arbitrary, this contradicts our supposition and proves the result. □

Proof of Theorem 5.2. By Lemma 2.2, we may assume that B is ordered. By Lemma 5.3 and a union bound,

$$\mathop{\mathbb{P}}_{T}\left[\forall k \leq k' < 2k\ \forall 1 \leq i \leq j \leq n\ \ L_2^{k'}(B_{i\cdots j}|_T) \leq \frac{1}{n}\right] \geq 1 - n^4 \cdot \frac{2}{2^k}.$$

Lemma 5.4 thus implies that

$$\mathop{\mathbb{P}}_{T}\left[\forall k'' \geq k\ \forall 1 \leq i \leq j \leq n\ \ L_2^{k''}(B_{i\cdots j}|_T) \leq \frac{1}{n}\right] \geq 1 - n^4 \cdot \frac{2}{2^k}.$$

Thus

$$\mathop{\mathbb{P}}_{T}\left[\sum_{k''\geq k} L_2^{k''}(B|_T) \leq 1\right] \geq 1 - n^4 \cdot \frac{2}{2^k}.$$

By Theorem 3.2,

$$\sum_{0<k'<k} L_2^{k'}(B|_T) \leq \sum_{0<k'<k} L_2^{k'}(B) \leq \sum_{0\leq k'\leq k-1} (2w^2)^{k'} = \frac{(2w^2)^k - 1}{2w^2 - 1} \leq (2w^2)^k - 1.$$

The result now follows. □

6 The Pseudorandom Generator

Theorem 6.1 (Main Result)**.** *There exists a pseudorandom generator family $G_{n,w,\varepsilon} : \{0,1\}^{s_{n,w,\varepsilon}} \to \{0,1\}^n$ with seed length*

$$s_{n,w,\varepsilon} = O(w^2 \log(w) \log(n) \log(nw/\varepsilon) + w^4 \log^2(w/\varepsilon))$$

such that, for any length-n, width-w, read-once, oblivious (but unordered), permutation branching program B and $\varepsilon > 0$,

$$\left|\left|\mathop{\mathbb{E}}_{U_{s_{n,w,\varepsilon}}}\left[B[G_{n,w,\varepsilon}(U_{s_{n,w,\varepsilon}})]\right] - \mathop{\mathbb{E}}_{U}\left[B[U]\right]\right|\right|_2 \leq \varepsilon.$$

Moreover, $G_{n,w,\varepsilon}$ can be computed in space $O(s_{n,w,\varepsilon})$.

The following lemma gives the basis of our pseudorandom generator.

Lemma 6.2. *Let B be a length-n, width-w, read-once, oblivious, regular branching program. Let $\varepsilon \in (0,1)$. Let T be a random variable over $\{0,1\}^n$ that is δ-almost $2k$-wise independent and each bit has expectation p, where we require*

$$p \le 1/4w^2, \qquad k \ge \log_2\left(4\sqrt{w}n^4/\varepsilon\right), \qquad \text{and} \qquad \delta \le (2w)^{-4k}.$$

Let U be uniform over $\{0,1\}^n$. Let X be a μ-biased random variable over $\{0,1\}^n$ with $\mu \le \varepsilon(2w^2)^{-k}$. Then

$$\left\| \mathop{\mathbb{E}}_{T,X,U}\left[B[\mathrm{Select}(T,X,U)]\right] - \mathop{\mathbb{E}}_{U}\left[B[U]\right] \right\|_2 \le 2\varepsilon.$$

Theorem 5.2 says that with high probability over T, $B|_T = \mathop{\mathbb{E}}_{U}\left[B[\mathrm{Select}(T,\cdot,U)]\right]$ has small Fourier mass. This implies that $B|_T$ is fooled by small bias X and thus

$$\mathop{\mathbb{E}}_{T,X,U}\left[B[\mathrm{Select}(T,X,U)]\right] \approx \mathop{\mathbb{E}}_{T,U,U'}\left[B[\mathrm{Select}(T,U',U)]\right] = \mathop{\mathbb{E}}_{U}\left[B[U]\right].$$

Proof. For $t \in \{0,1\}^n$, we have

$$\begin{aligned}
\left\| \mathop{\mathbb{E}}_{X,U}\left[B[\mathrm{Select}(t,X,U)]\right] - \mathop{\mathbb{E}}_{U}\left[B[U]\right] \right\|_2 &= \left\| \mathop{\mathbb{E}}_{X}\left[B|_t[X]\right] - \mathop{\mathbb{E}}_{U}\left[B[U]\right] \right\|_2 \\
&= \left\| \sum_{s\neq 0} \widehat{B|_t}[s]\widehat{X}(s) \right\|_2 \\
&\le \sum_{s\neq 0} \left\| \widehat{B|_t}[s] \right\|_2 |\widehat{X}(s)| \\
&\le L_2(B|_t)\mu.
\end{aligned}$$

We apply Theorem 5.2 and, with probability at least $1 - 2\cdot n^4/2^k$ over T, we have $L_2(B|_T)\mu \le \varepsilon$. Thus

$$\begin{aligned}
\left\| \mathop{\mathbb{E}}_{T,X,U}\left[B[\mathrm{Select}(T,X,U)]\right] - \mathop{\mathbb{E}}_{U}\left[B[U]\right] \right\|_2 &\le \mathop{\mathbb{P}}_{T}\left[L_2(B|_T)\mu > \varepsilon\right] \max_t \left\| \mathop{\mathbb{E}}_{X,U}\left[B[\mathrm{Select}(t,X,U)]\right] - \mathop{\mathbb{E}}_{U}\left[B[U]\right] \right\|_2 \\
&\quad + \mathop{\mathbb{P}}_{T}\left[L_2(B|_T)\mu \le \varepsilon\right]\varepsilon \\
&\le 2n^4/2^k \cdot 2\sqrt{w} + \varepsilon \\
&\le 2\varepsilon.
\end{aligned}$$

□

Now we use the above results to construct our pseudorandom generator for a read-once, oblivious, permutation branching program B.

Lemma 6.2 says that, if we define $\overline{B}_{t,x}[y] := B[\mathrm{Select}(t,x,y)]$, then $\mathop{\mathbb{E}}_{T,X,U}\left[\overline{B}_{T,X}[U]\right] \approx \mathop{\mathbb{E}}_{U}\left[B[U]\right]$, where T is almost k-wise independent with each bit having expectation p and X has small bias. So now we need only construct a pseudorandom generator for $\overline{B}_{t,x}$, which is a length-$(n-|t|)$ permutation branching program. Then

$$\mathop{\mathbb{E}}_{T,X,\tilde{U}}\left[\overline{B}_{T,X}[\tilde{U}]\right] \approx \mathop{\mathbb{E}}_{T,X,U}\left[\overline{B}_{T,X}[U]\right] \approx \mathop{\mathbb{E}}_{U}\left[B[U]\right],$$

where $\tilde{U}$ is the pseudorandom generator for $\overline{B}_{t,x}$. We construct $\tilde{U} \in \{0,1\}^{n-|T|}$ recursively; each time we recurse, the required seed length is reduced to $n - |T| \approx n(1-p)$. Thus after $O(\log(n)/p)$ levels of recursion the required seed length is constant.

The only place where the analysis breaks down for regular branching programs is when we recurse. If B is only a *regular* branching program, $\overline{B}_{t,x}$ may not be regular. However, if B is a *permutation* branching program, then $\overline{B}_{t,x}$ is too. Essentially, the only obstacle to generalising the analysis to regular branching programs is that regular branching programs are not closed under restrictions.

The pseudorandom generator is formally defined as follows.

Algorithm for $G_{n,w,\varepsilon} : \{0,1\}^{s_{n,w,\varepsilon}} \to \{0,1\}^n$.

Parameters: $n \in \mathbb{N}$, $w \in \mathbb{N}$, $\varepsilon > 0$.

Input: A random seed of length $s_{n,w,\varepsilon}$.

1. Compute appropriate values of $p \in [1/8w^2, 1/4w^2]$, $k \geq \log_2\left(4\sqrt{w}n^4/\varepsilon\right)$, $\delta = \varepsilon(2w)^{-4k}$, and $\mu = \varepsilon(2w^2)^{-k}$.[1]
2. If $n \leq (4 \cdot \log_2(2/\varepsilon)/p)^2$, output n truly random bits and stop.
3. Sample $T \in \{0,1\}^n$ where each bit has expectation p and the bits are δ-almost $2k$-wise independent.
4. If $|T| < pn/2$, output 0^n and stop.
5. Recursively sample $\tilde{U} \in \{0,1\}^{\lfloor n(1-p/2)\rfloor}$. i.e. $\tilde{U} = G_{\lfloor n(1-p/2)\rfloor,w,\varepsilon}(U)$.
6. Sample $X \in \{0,1\}^n$ from a μ-biased distribution.
7. Output $\text{Select}(T, X, \tilde{U}) \in \{0,1\}^n$.

The analysis of the algorithm proceeds roughly as follows.

- Every time we recurse, n is decreased to $\lfloor n(1-p/2)\rfloor$. After $O(\log(n)/p)$ recursions, n is reduced to $O(1)$. So the maximum recursion depth is $r = O(\log(n)/p)$.
- The probability of failing because $|T| < pn/2$ is small by a Chernoff bound for limited independence. (This requires that n is not too small and, hence, step 2.)
- The output is pseudorandom, as

$$\mathop{\mathbb{E}}_{U}\left[B[G_{n,w,\varepsilon}(U)]\right] = \mathop{\mathbb{E}}_{T,X,\tilde{U}}\left[B[\text{Select}(T,X,\tilde{U})]\right] \approx \mathop{\mathbb{E}}_{T,X,U}\left[B[\text{Select}(T,X,U)]\right] \approx \mathop{\mathbb{E}}_{U}\left[B[U]\right].$$

  The first approximate equality holds because we inductively assume that $\tilde{U}$ is pseudorandom. The second approximate equality holds by Lemma 6.2.
- The total seed length is the seed length needed to sample X and T at each level of recursion and $O((\log(1/\varepsilon)/p)^2)$ truly random bits at the last level. Sampling X requires seed length $O(\log(n/\mu)) = O(\log(n/\varepsilon)+k\log(w))$ and sampling T requires seed length $O(k\log(1/p)+\log(\log(n)/\delta)) = O(k\log(w)+\log(\log(n)/\varepsilon))$ so the total seed length is

$$O(r \cdot (k\log(w) + \log(n/\varepsilon)) + w^4\log^2(1/\varepsilon)) = O(w^2\log(w)\log(n)\log(nw/\varepsilon) + w^4\log^2(1/\varepsilon)).$$

Lemma 6.3. *The probability that $G_{n,w,\varepsilon}$ fails at step 4 is bounded by 2ε—that is, $\mathop{\mathbb{P}}_{T}[|T| < pn/2] \leq 2\varepsilon$.*

[1] For the purposes of the analysis we assume that p, k, δ, and μ are the same at every level of recursion. So if $G_{n,w,\varepsilon}$ is being called recursively, use the same values of p, k, δ, and μ as at the previous level of recursion.

Proof. By a Chernoff bound for limited independence (see Lemma B.1),

$$\mathop{\mathbb{P}}_{T}[|T| < pn/2] \le \left(\frac{k'^2}{4n(p/2)^2}\right)^{\lfloor k'/2 \rfloor} + \frac{\delta}{(p/2)^{k'}},$$

where $k' \le 2k$ is arbitrary. Set $k' = 2\lceil \log_2(1/\varepsilon) \rceil$. Step 2 ensures that $n > (4 \cdot \log_2(2/\varepsilon)/p)^2 > (2k'/p)^2$. Thus we have

$$\mathop{\mathbb{P}}_{T}[|T| < pn/2] \le \left(\frac{k'^2}{4(2k'/p)^2(p/2)^2}\right)^{\log_2(1/\varepsilon)} + \frac{\varepsilon(2w)^{-4k}}{(p/2)^k} \le 2\varepsilon.$$

□

The following bounds the error of $G_{n,w,\varepsilon}$.

Lemma 6.4. *Let B be a length-n, width-w, read-once, oblivious, permutation branching program. Then*

$$\left\| \mathop{\mathbb{E}}_{U_{s_{n,w,\varepsilon}}} \left[B[G_{n,w,\varepsilon}(U_{s_{n,w,\varepsilon}})]\right] - \mathop{\mathbb{E}}_{U}[B[U]] \right\|_2 \le 6\sqrt{w} r \varepsilon = O(w^{2.5}\log(n)\varepsilon),$$

where $r = O(\log(n)/p)$ is the maximum recursion depth of $G_{n,w,\varepsilon}$.

Proof. For $0 \le i < r$, let n_i, T_i, X_i, and $\tilde{U}_i$ be the values of n, T, X, and $\tilde{U}$ at recursion level i. We have $n_{i+1} = \lfloor n_i(1-p/2) \rfloor \le n(1-p/2)^{i+1}$ and $\tilde{U}_{i-1} = \mathrm{Select}(T_i, X_i, \tilde{U}_i)$. Let Δ_i be the error of the output from the i^{th} level of recursion—that is,

$$\Delta_i := \max_{B'} \left\| \mathop{\mathbb{E}}_{T_i, X_i, \tilde{U}_i} \left[B'[\mathrm{Select}(T_i, X_i, \tilde{U}_i)]\right] - \mathop{\mathbb{E}}_{U}[B'[U]] \right\|_2,$$

where the maximum is taken over all length-n_i, width-w, read-once, oblivious, permutation branching programs B'.

Since the last level of recursion outputs uniform randomness, $\Delta_r = 0$. For $0 \le i < r$, we have, for some B',

$$\begin{aligned}
\Delta_i &\le \left\| \mathop{\mathbb{E}}_{T_i, X_i, \tilde{U}_i} \left[B'[\mathrm{Select}(T_i, X_i, \tilde{U}_i)]\right] - \mathop{\mathbb{E}}_{U}[B'[U]] \right\|_2 \cdot \mathop{\mathbb{P}}_{T}[|T| \ge pn/2] \\
&\quad + 2\sqrt{w} \cdot \mathop{\mathbb{P}}_{T}[|T| < pn/2] \\
&\le \left\| \mathop{\mathbb{E}}_{T_i, X_i, \tilde{U}_i} \left[B'[\mathrm{Select}(T_i, X_i, \tilde{U}_i)]\right] - \mathop{\mathbb{E}}_{T_i, X_i, U}[B'[\mathrm{Select}(T_i, X_i, U)]] \right\|_2 + \left\| \mathop{\mathbb{E}}_{T_i, X_i, U}[B'[\mathrm{Select}(T_i, X_i, U)]] - \mathop{\mathbb{E}}_{U}[B'[U]] \right\|_2 \\
&\quad + 2\sqrt{w} \cdot \mathop{\mathbb{P}}_{T}[|T| < pn/2]
\end{aligned}$$

By Lemma 6.2,

$$\left\| \mathop{\mathbb{E}}_{T_i, X_i, U}[B'[\mathrm{Select}(T_i, X_i, U)]] - \mathop{\mathbb{E}}_{U}[B'[U]] \right\|_2 \le 2\varepsilon.$$

By Lemma 6.3,

$$\mathop{\mathbb{P}}_{T}[|T| < pn/2] \le 2\varepsilon.$$

We claim that

$$\left\| \mathop{\mathbb{E}}_{T_i, X_i, \tilde{U}_i} \left[B'[\mathrm{Select}(T_i, X_i, \tilde{U}_i)]\right] - \mathop{\mathbb{E}}_{T_i, X_i, U}[B'[\mathrm{Select}(T_i, X_i, U)]] \right\|_2 \le \Delta_{i+1}.$$

Before we prove the claim, we complete the proof: This gives $\Delta_i \le \Delta_{i+1} + 2\varepsilon + 2\sqrt{w} \cdot 2\varepsilon$. It follows that $\Delta_0 \le 6\sqrt{w}r\varepsilon$, as required.

Now to prove the claim. In fact, we prove a stronger result: for *every* fixed $T_i = t$ and $X_i = x$. We have

$$\left|\left|\mathop{\mathbb{E}}_{\tilde{U}_i}\left[B'[\text{Select}(t,x,\tilde{U}_i)]\right] - \mathop{\mathbb{E}}_{U}\left[B'[\text{Select}(t,x,U)]\right]\right|\right|_2 \le \Delta_{i+1}.$$

Consider $\overline{B}_{x,t}[y] := B'[\text{Select}(t,x,y)]$ as a function of $y \in \{0,1\}^{n_i - |t|}$. Then $\overline{B}_{x,t}$ is a width-w, length-$(n_i - |t|)$, read-once, oblivious, permutation branching program—$\overline{B}_{x,t}$ is obtained from B' by fixing the bits in t to the values from x and 'collapsing' those layers. (If B' is a *regular* branching program, then $\overline{B}_{x,t}$ is not necessarily a regular branching program. This is the only part of the proof where we need to assume that B is a permutation branching program.)

We inductively know that $\tilde{U}_i$ is pseudorandom for $\overline{B}_{x,t}$—that is, $\left|\left|\mathop{\mathbb{E}}_{\tilde{U}_i}\left[\overline{B}_{x,t}[\tilde{U}_i]\right] - \mathop{\mathbb{E}}_{U}\left[\overline{B}_{x,t}[U]\right]\right|\right|_2 \le \Delta_{i+1}$. Thus

$$\left|\left|\mathop{\mathbb{E}}_{\tilde{U}_i}\left[B'[\text{Select}(t,x,\tilde{U}_i)]\right] - \mathop{\mathbb{E}}_{U}\left[B'[\text{Select}(t,x,U)]\right]\right|\right|_2 = \left|\left|\mathop{\mathbb{E}}_{\tilde{U}_i}\left[\overline{B}_{x,t}[\tilde{U}_i]\right] - \mathop{\mathbb{E}}_{U}\left[\overline{B}_{x,t}[U]\right]\right|\right|_2 \le \Delta_{i+1},$$

as required. □

Proof of Theorem 6.1. Choose $\varepsilon' = \Theta(\varepsilon/w^{2.5}\log(n))$ such that $G_{n,w,\varepsilon'}$ has error ε. The seed length is

$$s_{n,w,\varepsilon'} = O(w^2\log(w)\log(n)\log(nw/\varepsilon) + w^4\log^2(w\log(n)/\varepsilon)),$$

as required. □

7 General Read-Once, Oblivious Branching Programs

With a different setting of parameters, our pseudorandom generator can fool arbitrary oblivious, read-once branching programs, rather than just permutation branching programs.

Theorem 7.1. *There exists a pseudorandom generator family $G'_{n,w,\varepsilon} : \{0,1\}^{s'_{n,w,\varepsilon}} \to \{0,1\}^n$ with seed length $s'_{n,w,\varepsilon} = O(\sqrt{n}\log^3(n)\log(nw/\varepsilon))$ such that, for any length-n, width-w, oblivious, read-once branching program B and $\varepsilon > 0$,*

$$\left|\left|\mathop{\mathbb{E}}_{U_{s'_{n,w,\varepsilon}}}\left[B[G'_{n,w,\varepsilon}(U_{s'_{n,w,\varepsilon}})]\right] - \mathop{\mathbb{E}}_{U}\left[B[U]\right]\right|\right|_2 \le \varepsilon.$$

Moreover, $G'_{n,w,\varepsilon}$ is computable in space $O(s_{n,w,\varepsilon})$.

Compared to Theorem 6.1, the seed length has a worse dependence on the length ($\sqrt{n}$ versus $\log^2 n$), but has a much better dependence on the width ($\log w$ versus $\text{poly}(w)$).

Impagliazzo, Meka, and Zuckerman [13] obtain the seed length $\sqrt{s} \cdot 2^{O(\sqrt{\log s})} = s^{1/2+o(1)}$ for arbitrary branching programs of size s. For a width-w, length-n, read-once branching program, $s = O(wn)$. Our result is incomparable to that of Impagliazzo et al. Our result only covers oblivious, read-once branching programs, while that of Impagliazzo et al. covers non-read-once and non-oblivious branching programs. However, our seed length depends logarithmically on the width, while theirs depends polynomially on the width; we can achieve seed length $n^{1/2+o(1)}$ for width $2^{n^{o(1)}}$ while they require width $n^{o(1)}$.

The key to proving Theorem 7.1 is the following Fourier mass bound for arbitrary branching programs.

Lemma 7.2. *Let B be a length-n, width-w, read-once, oblivious branching program. Then, for all $k \in [n]$,*

$$L_2^k(B) := \sum_{s \in \{0,1\}^n : |s| = k} \left|\left|\widehat{B}[s]\right|\right|_2 \leq \sqrt{w\binom{n}{k}} \leq \sqrt{wn^k}.$$

Proof. By Parseval's Identity,

$$\sum_{s \in \{0,1\}^n : |s| = k} \left|\left|\widehat{B}[s]\right|\right|_2^2 \leq \sum_{s \in \{0,1\}^n} \left|\left|\widehat{B}[s]\right|\right|_{\mathrm{Fr}}^2 = \mathop{\mathbb{E}}_{U}\left[||B[U]||_{\mathrm{Fr}}^2\right] = w.$$

The result follows from Cauchy-Schwartz. □

The bound of Lemma 7.2 is different that of Theorem 3.2. This leads to the different seed length in Theorem 7.1 versus Theorem 6.1.

Lemma 7.2 gives a different version of our main lemma (Theorem 5.2).

Lemma 7.3. *Let B be a length-n, width-w, read-once, oblivious branching program. Let T be a random variable over $\{0,1\}^n$ where each bit has expectation p and the bits are $2k$-wise independent. Suppose $p \leq 1/\sqrt{4n}$. Then*

$$\mathop{\mathbb{P}}_{T}\left[L_2(B|_T) \leq \sqrt{w} \cdot n^{k/2}\right] \geq 1 - k \cdot \sqrt{w} \cdot n^3 \cdot 2^{-k}.$$

Using Lemma 7.3, we can construct a pseudorandom generator fooling general read-once, oblivious branching programs (Theorem 7.1), similarly to the proof of Theorem 6.1. For more details, see Appendix C.

8 Further Work

One open problem is to extend the main result (Theorem 6.1) to regular or even non-regular branching programs while maintaining polylog(n) seed length. As discussed in Section 6, the only part of our analysis that fails for regular branching programs is the recursive analysis. The problem is that regular branching programs are not closed under restriction—that is, setting some of the bits of a regular branching program does not necessarily yield a regular branching program. In particular, we cannot bound

$$\left|\left|\mathop{\mathbb{E}}_{\tilde{U}}\left[B[\mathrm{Select}(t,x,\tilde{U})]\right] - \mathop{\mathbb{E}}_{U}\left[B[\mathrm{Select}(t,x,U)]\right]\right|\right|$$

for fixed t and x by the distinguishability of $\tilde{U}$ and U by another read-once, oblivious, regular branching program $\overline{B}_{x,t}$. We have two options:

- Find another way to bound $\left|\left|\mathop{\mathbb{E}}_{T,X,\tilde{U}}\left[B[\mathrm{Select}(T,X,\tilde{U})]\right] - \mathop{\mathbb{E}}_{T,X,}\left[B[\mathrm{Select}(T,X,U)]\right]\right|\right|$.
- Extend the main lemma (Theorem 5.2) to non-regular branching programs.

Towards the latter option, we have the following conjecture.

Conjecture 8.1. *For every constant w, the following holds. Let B be a length-n, width-w, read-once, oblivious branching program. Then*

$$L_2^k(B) = \sum_{s \in \{0,1\}^n : |s| = k} \left|\left|\widehat{B}[s]\right|\right|_2 \leq n^{O(1)}(\log n)^{O(k)}$$

for all $k \geq 1$.

This conjecture relates to the Coin Theorem of Brody and Verbin (see the discussion in Section 4). Specifically, if we remove the $n^{O(1)}$ factor, this conjecture implies the Coin Theorem.

Conjecture 8.1 would suffice to construct a pseudorandom generator for constant-width, read-once, oblivious branching programs with seed length $\text{polylog}(n)$.

The seed length of our generators is worse than that of generators for ordered branching programs. Indeed, for ordered *permutation* branching programs of constant width, it is known how to achieve seed length $O(\log n)$ [17], whereas we only achieve seed length $O(\log^2 n)$ in Theorem 6.1. For general ordered branching programs, Nisan [21] obtains seed length $O(\log(nw)\log(n))$, whereas Theorem 7.1 gives seed length $\tilde{O}(\sqrt{n}\log(w))$. It would be interesting to close these gaps.

References

[1] Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple constructions of almost k-wise independent random variables. In *FOCS*, pages 544–553, 1990.

[2] Roy Armoni, Michael E. Saks, Avi Wigderson, and Shiyu Zhou. Discrepancy sets and pseudorandom generators for combinatorial rectangles. In *FOCS*, pages 412–421, 1996.

[3] Andrej Bogdanov, Zeev Dvir, Elad Verbin, and Amir Yehudayoff. Pseudorandomness for width 2 branching programs. *Electronic Colloquium on Computational Complexity (ECCC)*, 16:70, 2009.

[4] Andrej Bogdanov, Periklis A. Papakonstantinou, and Andrew Wan. Pseudorandomness for read-once formulas. In *FOCS*, pages 240–246, 2011.

[5] Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. Pseudorandom generators for regular branching programs. *Foundations of Computer Science, IEEE Annual Symposium on*, 0:40–47, 2010.

[6] Joshua Brody and Elad Verbin. The coin problem and pseudorandomness for branching programs. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 30–39, Washington, DC, USA, 2010. IEEE Computer Society.

[7] L. Elisa Celis, Omer Reingold, Gil Segev, and Udi Wieder. Balls and bins: Smaller hash families and faster evaluation. In *FOCS*, pages 599–608, 2011.

[8] Anindya De. Pseudorandomness for permutation and regular branching programs. In *Proceedings of the 2011 IEEE 26th Annual Conference on Computational Complexity*, CCC '11, pages 221–231, Washington, DC, USA, 2011. IEEE Computer Society.

[9] Guy Even, Oded Goldreich, Michael Luby, Noam Nisan, and Boban Velickovic. Efficient approximation of product distributions. *Random Struct. Algorithms*, 13(1):1–16, 1998.

[10] Parikshit Gopalan, Raghu Meka, Omer Reingold, Luca Trevisan, and Salil Vadhan. Better pseudorandom generators from milder pseudorandom restrictions. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 120–129, 2012.

[11] Iftach Haitner, Danny Harnik, and Omer Reingold. On the power of the randomized iterate. In C. Dwork, editor, *Advances in Cryptology—CRYPTO '06*, Lecture Notes in Computer Science. Springer-Verlag, 2006.

[12] Alexander Healy, Salil Vadhan, and Emanuele Viola. Using nondeterminism to amplify hardness. *SIAM Journal on Computing*, 35(4):903–931 (electronic), 2006.

[13] R. Impagliazzo, R. Meka, and D. Zuckerman. Pseudorandomness from shrinkage. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 111–119, 2012.

[14] Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *In Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 356–364, 1994.

[15] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, 2006.

[16] Eyal Kaplan, Moni Naor, and Omer Reingold. Derandomized constructions of k-wise (almost) independent permutations. In *Proceedings of the 8th International Workshop on Randomization and Computation (RANDOM '05)*, number 3624 in Lecture Notes in Computer Science, pages 354 – 365, Berkeley, CA, August 2005. Springer.

[17] Michal Koucký, Prajakta Nimbhorkar, and Pavel Pudlák. Pseudorandom generators for group products. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, STOC '11, pages 263–272, New York, NY, USA, 2011. ACM.

[18] Nathan Linial, Michael Luby, Michael E. Saks, and David Zuckerman. Efficient construction of a small hitting set for combinatorial rectangles in high dimension. *Combinatorica*, 17(2):215–234, 1997.

[19] Chi-Jen Lu. Improved pseudorandom generators for combinatorial rectangles. *Combinatorica*, 22(3):417–434, 2002.

[20] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput*, 22:838–856, 1993.

[21] Noam Nisan. $\mathcal{RL} \subset \mathcal{SC}$. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, STOC '92, pages 619–623, New York, NY, USA, 1992. ACM.

[22] Noam Nisan and David Zuckerman. More deterministic simulation in logspace. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, STOC '93, pages 235–244, New York, NY, USA, 1993. ACM.

[23] Ran Raz and Omer Reingold. On recycling the randomness of states in space bounded computation. In *In Proceedings of the Thirty-First Annual ACM Symposium on the Theory of Computing*, pages 159–168, 1999.

[24] Omer Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4):17:1–17:24, September 2008.

[25] Omer Reingold, Luca Trevisan, and Salil Vadhan. Pseudorandom walks on regular digraphs and the $\mathcal{RL}$ vs. $\mathcal{L}$ problem. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, STOC '06, pages 457–466, New York, NY, USA, 2006. ACM.

[26] Eyal Rozenman and Salil Vadhan. Derandomized squaring of graphs. In *Proceedings of the 8th international workshop on Approximation, Randomization and Combinatorial Optimization Problems, and Proceedings of the 9th international conference on Randamization and Computation: algorithms and techniques*, APPROX'05/RANDOM'05, pages 436–447, Berlin, Heidelberg, 2005. Springer-Verlag.

[27] Michael Saks and Shiyu Zhou. $\mathrm{BP_HSPACE}(S) \subset \mathrm{DSPACE}(S^{3/2})$. *Journal of Computer and System Sciences*, 58(2):376 – 403, 1999.

[28] J. Schmidt, A. Siegel, and A. Srinivasan. ChernoffHoeffding bounds for applications with limited independence. *SIAM Journal on Discrete Mathematics*, 8(2):223–250, 1995.

[29] D. Sivakumar. Algorithmic derandomization via complexity theory. In *IEEE Conference on Computational Complexity*, page 10, 2002.

[30] Thomas Steinke. Pseudorandomness for permutation branching programs without the group theory. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:83, 2012.

[31] Yoav Tzur. Notions of weak pseudorandomness and $\mathrm{GF}(2^n)$-polynomials. Master's thesis, Weizmann Institute of Science, 2009.

[32] Jiří Šíma and Stanislav Žák. A sufficient condition for sets hitting the class of read-once branching programs of width 3. In *Proceedings of the 38th international conference on Current Trends in Theory and Practice of Computer Science*, SOFSEM'12, pages 406–418, Berlin, Heidelberg, 2012. Springer-Verlag.

A Proof of Lemma 3.1

Lemma 3.1. *Let B be a length-n, width-w, ordered, regular branching program. Then*

$$\sum_{1 \le i \le n} \left|\left| \widehat{B_{i \cdots n}[1 \circ 0^{n-i}]} \right|\right|_2 \le 2w^2.$$

This proof is adapted from [5, Lemma 4].

Proof. Define $\rho : \mathbb{R}^{w \times w} \to \mathbb{R}$ by

$$\rho(X) := \sum_{1 \le u < v \le w} ||X(u, \cdot) - X(v, \cdot)||_2 ,$$

where $X(u,\cdot)$ and $X(v,\cdot)$ are the u^{th} and v^{th} rows of X respectively. We claim that, for all $i \in [n]$ and $X \in \mathbb{R}^{w\times w}$,

$$\left|\left|\widehat{B_i}[1]X\right|\right|_2 \leq 2(\rho(X) - \rho(\widehat{B_i}[0]X)).$$

It follows that

$$\sum_{i\in[n]} \left|\left|\widehat{B_i}[1]\widehat{B_{i+1\cdots n}}[0^{n-i}]\right|\right|_2 \leq 2\sum_{i\in[n]} \rho(\widehat{B_{i+1\cdots n}}[0^{n-i}]) - \rho(\widehat{B_{i\cdots n}}[0^{n-i+1}]) = 2(\rho(I) - \rho(\widehat{B}[0^n])).$$

Noting that $\rho(I) \leq w^2$, we obtain the result.

Intuitively, $\rho(\widehat{B_{i\cdots n}}[0^n])$ measures the 'correlation' between the state at layer i and the state at the last layer when run on uniform randomness; ρ measures how much the distribution of the final state changes if the state at layer i is changed from u to v and this is summed over all pairs $\{u, v\}$. On the other hand, $\left|\left|\widehat{B_{i\cdots n}}[1 \circ 0^{n-i}]\right|\right|_2$ measures the correlation between bit i and the final state; the Fourier coefficient shows how much $B_{i\cdots n}$ correlates with bit i. Our claim simply states that this correlation is 'conserved'—that is, the correlation of the state at layer i plus the correlation of bit i is bounded by the correlation of the state at layer $i+1$. This makes sense as the state at layer $i+1$ is determined by the state at layer i and bit i.

Now we prove our claim: Consider the rows of X and $\widehat{B_i}[0]X$ as points in $\mathbb{R}^w$. We start with $2w$ points corresponding to each row of X repeated twice and we move these points one by one until they correspond to the rows of $\widehat{B_i}[0]X$ repeated twice: In step $u \in [w]$, we take one point corresponding to row $B_i[0](u)$ of X (that is, $B_i[0](u,\cdot)X$) and one point corresponding to row $B_i[1](u)$ of X ($B_i[1](u,\cdot)X$) and move both to their midpoint $\widehat{B_i}[0](u,\cdot)X = (B_i[0](u,\cdot)X + B_i[1](u,\cdot)X)/2$.

Let P_u be the multiset of points at step $u \in [w] \cup \{0\}$. That is,

$$P_0 = \bigcup_{u\in[w]} \{X(u,\cdot), X(u,\cdot)\} \quad \text{and} \quad \forall u \in [w]\ \ P_u = (P_{u-1}\backslash\{B_i[0](u,\cdot)X, B_i[1](u,\cdot)X\}) \cup \{\widehat{B_i}[0](u,\cdot)X, \widehat{B_i}[0](u,\cdot)X\}.$$

By regularity, we have $P_w = \bigcup_{u\in[w]} \{\widehat{B_i}[0](u,\cdot)X, \widehat{B_i}[0](u,\cdot)X\}$.

Now we can consider ρ as a function on the collection of points P_u:

$$\rho'(P_u) := \sum_{x,y\in P_u : x \prec y} ||x - y||_2 ,$$

where the comparison $x \prec y$ is with respect to some arbitrary ordering on the multiset of points. (We simply do not want to double count pairs.) We have $\rho'(P_0) = 4\rho(X)$ and $\rho'(P_w) = 4\rho(\widehat{B_i}[0]X)$. Note that the 4 factor comes from the fact that every pair of rows $(X(u,\cdot), X(v,\cdot))$ becomes four pairs of points in P_0, as every row corresponds to two points.

Fix $u \in [w]$. Now we bound $\rho'(P_{u-1}) - \rho'(P_u)$: Let $x = B_i[0](u,\cdot)X$, $y = B_i[1](u,\cdot)X$, and $z = \widehat{B_i}[0](u,\cdot)X$. Then $P_u = (P_{u-1}\backslash\{x, y\}) \cup \{z, z\}$ and $z = (x + y)/2$. We have

$$\rho'(P_{u-1}) - \rho'(P_u) = ||x - y||_2 + \sum_{w\in(P_{u-1}\backslash\{x,y\})} ||w - x||_2 + ||w - y||_2 - 2\,||w - z||_2 .$$

By the triangle inequality,

$$2\,||w - z||_2 = ||2w - (x + y)||_2 \leq ||w - x||_2 + ||w - y||_2 .$$

So

$$\rho'(P_{u-1}) - \rho'(P_u) \geq ||x - y||_2 = ||B_i[0](u,\cdot)X - B_i[1](u,\cdot)X||_2 = \left|\left|2\widehat{B_i}[1](u,\cdot)X\right|\right|_2 .$$

It follows that

$$4\rho(X) - 4\rho(\widehat{B_i}[0]X) = \rho'(P_0) - \rho'(P_w) = \sum_{u\in[w]} \rho'(P_{u-1}) - \rho'(P_u) \geq 2\sum_{u\in[w]} \left|\left|\widehat{B_i}[1](u,\cdot)X\right|\right|_2 \geq 2\left|\left|\widehat{B_i}[1]X\right|\right|_2.$$

□

B Limited Independence

We use the following fact.

Lemma B.1 (Chernoff Bound for Limited Independence)**.** *Let $X_1 \cdots X_\ell$ be δ-almost k-wise independent random variables with $0 \leq X_i \leq 1$ for all i. Set $X = \sum_i X_i$. Then, for all $\zeta \in (0,1)$,*

$$\mathop{\mathbb{P}}_X\left[\left|X - \mathop{\mathbb{E}}_X[X]\right| \geq \ell\zeta\right] \leq \left(\frac{k^2}{4\ell\zeta^2}\right)^{\lfloor k/2\rfloor} + \frac{\delta}{\zeta^k}$$

The following proof is based on [28, Theorem 4]. The only difference is that we extend to almost k-wise independence from k-wise independence.

Proof. Assume, without loss of generality, that k is even. Let $\mu = \mathop{\mathbb{E}}_X[X]$ and $\mu_i = \mathop{\mathbb{E}}_X[X_i]$. We have

$$\mathop{\mathbb{E}}_X\left[(X-\mu)^k\right] = \mathop{\mathbb{E}}_X\left[\left(\sum_{1\leq i\leq \ell} X_i - \mu_i\right)^k\right] = \mathop{\mathbb{E}}_X\left[\sum_{S\in[\ell]^k}\prod_{i\in S}(X_i-\mu_i)\right] = \sum_{S\in[\ell]^k}\mathop{\mathbb{E}}_X\left[\prod_{i\in S}(X_i-\mu_i)\right].$$

Note that $\mathop{\mathbb{E}}_X[X_i - \mu_i] = 0$ for all i. By δ-almost k-wise independence, each term in the product $\prod_{i\in S}(X_i - \mu_i)$ is almost independent unless it is a repeated term. Thus, unless every term is a repeated term, $\left|\mathop{\mathbb{E}}_X\left[\prod_{i\in S}(X_i-\mu_i)\right]\right| \leq \delta$. If every term is repeated, then there are at most $k/2$ different terms. This means we need only consider $\binom{\ell}{k/2}(k/2)^k$ values of S.

Also $|X_i - \mu_i| \leq 1$, so $\mathop{\mathbb{E}}_X\left[\prod_{i\in S}(X_i-\mu_i)\right] \leq 1$. Thus

$$\mathop{\mathbb{E}}_X\left[(X-\mu)^k\right] \leq \binom{\ell}{k/2}(k/2)^k + \ell^k\delta \leq \left((k/2)^2\ell\right)^{k/2} + \ell^k\delta.$$

And, by Markov's inequality,

$$\mathop{\mathbb{P}}_X\left[|X-\mu| \geq \ell\zeta\right] = \mathop{\mathbb{P}}_X\left[(X-\mu)^k \geq (\ell\zeta)^k\right] \leq \frac{\mathop{\mathbb{E}}_X\left[(X-\mu)^k\right]}{(\ell\zeta)^k} \leq \left(\frac{k^2}{4\ell\zeta^2}\right)^{k/2} + \frac{\delta}{\zeta^k},$$

as required. □

Lemma B.2. *We can sample a δ-almost k-wise independent random variable T over $\{0,1\}^n$ with each bit having expectation $p = 2^{-d}$ using $O(kd + \log(\log(n)/\delta))$ random bits.*

Proof. The algorithm is as follows.

1. Sample $X \in \{0,1\}^{nd}$ that is δ-almost kd-wise independent.

2. Sample $Y \in \{0,1\}^d$ uniformly at random.
3. Let $Z = X \oplus (Y, Y, \cdots, Y)$. That is, XOR X with n copies of Y.
4. Let $T(i) = \prod_{(i-1)d<j\leq id} Z(j)$ for all i.
5. Output T.

Sampling X requires $O(kd + \log(\log(n)/\delta))$ random bits [20, 1]. Sampling Y requires d random bits. By XORing, Z has the property that it is both δ-almost kd-wise independent and every block of d consecutive bits is independent. The latter property ensures that, for all i,

$$\mathop{\mathbb{P}}_{T}[T(i) = 1] = \mathop{\mathbb{P}}_{Z}[Z((i-1)d+1) = Z((i-1)d+2) = \cdots = Z(id) = 1] = 2^{-d} = p.$$

The former property ensures that T is δ-almost k-wise independent, as required. □

C General Read-Once, Oblivious Branching Programs

First we prove the main lemma for general branching programs:

Lemma 7.3. *Let B be a length-n, width-w, read-once, oblivious branching program. Let T be a random variable over $\{0,1\}^n$ where each bit has expectation p and the bits are $2k$-wise independent. Suppose $p \leq 1/\sqrt{4n}$. Then*

$$\mathop{\mathbb{P}}_{T}\left[L_2(B|_T) \leq \sqrt{w} \cdot n^{k/2}\right] \geq 1 - k \cdot \sqrt{w} \cdot n^3 \cdot 2^{-k}.$$

Proof. Fix i, j, and k' with $1 \leq i \leq j \leq n$ and $k \leq k' < 2k$. Then

$$\mathop{\mathbb{E}}_{T}\left[L_2^{k'}(B_{i\cdots j}|_T)\right] = \sum_{s\subset[n]:|s|=k'} \mathop{\mathbb{P}}_{T}[s \subset T] \left|\left|\widehat{B_{i\cdots j}}[s]\right|\right| \leq \sqrt{wn^{k'}}p^{k'} \leq \sqrt{w}2^{-k'} \leq \sqrt{w}2^{-k}.$$

By Markov's inequality and a union bound,

$$\mathop{\mathbb{P}}_{T}\left[\forall 1 \leq i \leq j \leq n\ \forall k \leq k' < 2k\ \ L_2^{k'}(B_{i\cdots j}|_T) \leq 1/n\right] \geq 1 - \sqrt{w} \cdot n^4 \cdot 2^{-k}.$$

Lemma 5.4 now implies that

$$\mathop{\mathbb{P}}_{T}\left[\forall 1 \leq i \leq j \leq n\ \forall k' \geq k\ \ L_2^{k'}(B_{i\cdots j}|_T) \leq 1/n\right] \geq 1 - \sqrt{w} \cdot n^4 \cdot 2^{-k}.$$

Thus

$$\mathop{\mathbb{P}}_{T}\left[\sum_{k'\geq k} L_2^{k'}(B|_T) \leq 1\right] \geq 1 - \sqrt{w} \cdot n^4 \cdot 2^{-k}.$$

Note that

$$\sum_{1\leq k'<k} L_2^{k'}(B|_T) \leq \sum_{1\leq k'<k} \sqrt{wn^{k'}} = \sqrt{wn}\frac{n^{(k-1)/2}-1}{\sqrt{n}-1} \leq \sqrt{w} \cdot n^{k/2} - 1.$$

The result follows. □

Now we prove the result for general read-once, oblivious branching programs:

Theorem 7.1. *There exists a pseudorandom generator family $G'_{n,w,\varepsilon} : \{0,1\}^{s'_{n,w,\varepsilon}} \to \{0,1\}^n$ with seed length $s'_{n,w,\varepsilon} = O(\sqrt{n}\log^3(n)\log(nw/\varepsilon))$ such that, for any length-n, width-w, oblivious, read-once branching program B and $\varepsilon > 0$,*

$$\left\| \mathop{\mathbb{E}}_{U_{s'_{n,w,\varepsilon}}} \left[B[G'_{n,w,\varepsilon}(U_{s'_{n,w,\varepsilon}})] \right] - \mathop{\mathbb{E}}_{U}[B[U]] \right\|_2 \leq \varepsilon.$$

Moreover, $G'_{n,w,\varepsilon}$ is computable in space $O(s_{n,w,\varepsilon})$.

The pseudorandom generator is formally defined as follows.

Algorithm for $G'_{n,w,\varepsilon} : \{0,1\}^{s'_{n,w,\varepsilon}} \to \{0,1\}^n$.

Parameters: $n \in \mathbb{N}$, $w \in \mathbb{N}$, $\varepsilon > 0$.

Input: A random seed of length $s'_{n,w,\varepsilon}$.

1. Compute appropriate values of $p \in [1/4\sqrt{n}, 1/2\sqrt{n}]$, $k \geq \log_2\left(2wn^4/\varepsilon\right)$, and $\mu = \varepsilon/\sqrt{wn^k}$.[2]
2. If $n \leq 16\log_2(2/\varepsilon)/p$, output n truly random bits and stop.
3. Sample $T \in \{0,1\}^n$ where each bit has expectation p and the bits are $2k$-wise independent.
4. If $|T| < pn/2$, output 0^n and stop.
5. Recursively sample $\tilde{U} \in \{0,1\}^{\lfloor n(1-p/2)\rfloor}$. i.e. $\tilde{U} = G'_{\lfloor n(1-p/2)\rfloor,w,\varepsilon}(U)$.
6. Sample $X \in \{0,1\}^n$ from a μ-biased distribution.
7. Output $\mathrm{Select}(T, X, \tilde{U}) \in \{0,1\}^n$.

Lemma C.1. *The probability that $G'_{n,w,\varepsilon}$ fails at step 4 is bounded by ε—that is, $\mathop{\mathbb{P}}_{T}[|T| < pn/2] \leq \varepsilon$.*

We need the following Chernoff bound for limited independence, which gives a better dependence on p than Lemma B.1.

Lemma C.2 ([28, Theorem 4 III])**.** *Let $X_1, X_2, \cdots, X_n$ be k-wise independent random variables on $[0,1]$. Let $X = \sum_i X_i$. Let $\mu = \mathop{\mathbb{E}}_{X}[X]$ and $\sigma^2 \geq Var[X]$. If $k \geq 2$ is even, then*

$$\mathop{\mathbb{P}}_{X}[|X - \mu| \geq \alpha] \leq \left(\frac{k \cdot \max\{k, \sigma^2\}}{e^{2/3}\alpha^2} \right)^{k/2}.$$

Proof of Lemma C.1. We have $\mathrm{Var}[|T|] = n\mathrm{Var}[T(i)] = np(1-p) \leq np$. By Lemma C.2,

$$\mathop{\mathbb{P}}_{T}[|T| < pn/2] \leq \left(\frac{2k'np}{e^{2/3}(pn/2)^2} \right)^{k'},$$

where $k' \leq k$ is arbitrary. Set $k' = \lceil \log_2(1/\varepsilon) \rceil$. Step 2 ensures that $n > 16\log_2(2/\varepsilon)/p > 16k'/p$. Thus we have

$$\mathop{\mathbb{P}}_{T}[|T| < pn/2] \leq \left(\frac{8k'}{e^{2/3}np} \right)^{\log_2(1/\varepsilon)} \leq \left(\frac{8k'}{e^{2/3}(16k')} \right)^{\log_2(1/\varepsilon)} \leq \varepsilon$$

. □

[2]For the purposes of the analysis we assume that p, k, and μ are the same at every level of recursion. So if $G'_{n,w,\varepsilon}$ is being called recursively, use the same values of p, k, and μ as at the previous level of recursion.

Lemma C.3. *Let B be a length-n, width-w, read-once, oblivious branching program. Let $\varepsilon \in (0,1)$. Let T be a random variable over $\{0,1\}^n$ that is $2k$-wise independent and each bit has expectation p, where we require*

$$p \le 1/\sqrt{4n}, \qquad k \ge \log_2\left(2wn^4/\varepsilon\right).$$

Let U be uniform over $\{0,1\}^n$. Let X be a μ-biased random variable over $\{0,1\}^n$ with $\mu \le \varepsilon/\sqrt{wn^k}$. Then

$$\left\| \mathop{\mathbb{E}}_{T,X,U}\left[B[\mathrm{Select}(T,X,U)]\right] - \mathop{\mathbb{E}}_{U}\left[B[U]\right] \right\|_2 \le 2\varepsilon.$$

Proof. For $t \in \{0,1\}^n$, we have

$$\left\| \mathop{\mathbb{E}}_{X,U}\left[B[\mathrm{Select}(t,X,U)]\right] - \mathop{\mathbb{E}}_{U}\left[B[U]\right] \right\|_2 = \left\| \mathop{\mathbb{E}}_{X}\left[B|_t[X]\right] - \mathop{\mathbb{E}}_{U}\left[B[U]\right] \right\|_2 \le L_2(B|_t)\mu.$$

We apply Lemma 7.3 and with probability at least $1 - \sqrt{w}n^4/2^k$ over T, we have $L_2(B|_T)\mu \le \varepsilon$. Thus

$$\begin{aligned}
\left\| \mathop{\mathbb{E}}_{T,X,U}\left[B[\mathrm{Select}(T,X,U)]\right] - \mathop{\mathbb{E}}_{U}\left[B[U]\right] \right\|_2 \le& \mathop{\mathbb{P}}_{T}\left[L_2(B|_T)\mu > \varepsilon\right] 2\sqrt{w} + \mathop{\mathbb{P}}_{T}\left[L_2(B|_T)\mu \le \varepsilon\right]\varepsilon \\
\le& \sqrt{w}n^4/2^k \cdot 2\sqrt{w} + \varepsilon \\
\le& 2\varepsilon.
\end{aligned}$$

□

Lemma C.4. *Let B be a length-n, width-w, read-once, oblivious branching program. Then*

$$\left\| \mathop{\mathbb{E}}_{U_{s'_{n,w,\varepsilon}}}\left[B[G'_{n,w,\varepsilon}(U_{s'_{n,w,\varepsilon}})]\right] - \mathop{\mathbb{E}}_{U}\left[B[U]\right] \right\|_2 \le 4\sqrt{w}r\varepsilon,$$

where $r = O(\log(n)/p)$ is the recursion depth of $G'_{n,w,\varepsilon}$.

The proof of this lemma is exactly as before (Lemma 6.4), except we no longer need the assumption that B is a permutation branching program.

Setting $\varepsilon' = O(\varepsilon/\sqrt{nw}\log(n))$ we can ensure that that $G'_{n,w,\varepsilon'}$ has error at most ε. The overall seed length is $O(\sqrt{n}\log^3(n)\log(nw/\varepsilon))$: Each of the $r = O(\sqrt{n}\log(n))$ levels of recursion requires $O(\log(w/\varepsilon) + k\log(n))$ random bits to sample X and $O(k\log^2 n)$ bits to sample T. Finally we need $O(\log(1/\varepsilon)/p)$ random bits at the last level. Since $k = O(\log(nw/\varepsilon))$, this gives the required seed length.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de