THEORY OF COMPUTING, Volume 13 (12), 2017, pp. 1–50
www.theoryofcomputing.org

SPECIAL ISSUE: APPROX-RANDOM 2014

Pseudorandomness and Fourier-Growth Bounds for Width-3 Branching Programs

Thomas Steinke[*] Salil Vadhan[†] Andrew Wan[‡]

Received February 23, 2015; Revised December 20, 2016; Published October 31, 2017

Abstract: We present an explicit pseudorandom generator for oblivious, read-once, width-3 branching programs, which can read their input bits in any order. The generator has seed length $\widetilde{O}(\log^3 n)$. The best previously known seed length for this model is $n^{1/2+o(1)}$ due to Impagliazzo, Meka, and Zuckerman (FOCS'12). Our result generalizes a recent result of Reingold, Steinke, and Vadhan (RANDOM'13) for *permutation* branching programs. The main technical novelty underlying our generator is a new bound on the Fourier growth of width-3, oblivious, read-once branching programs. Specifically, we show that for any $f : \{0,1\}^n \to \{0,1\}$ computed by such a branching program, and $k \in [n]$,

$$\sum_{s\subseteq[n]:|s|=k} \left|\widehat{f}[s]\right| \le n^2 \cdot (O(\log n))^k,$$

where $\widehat{f}[s] = \mathbb{E}_U\left[f[U]\cdot(-1)^{s\cdot U}\right]$ is the standard Fourier transform over $\mathbb{Z}_2^n$. The base $O(\log n)$ of the Fourier growth is tight up to a factor of $\log\log n$.

ACM Classification: F.1

AMS Classification: 65C10, 68Q87

Key words and phrases: pseudorandom generators, branching programs, Fourier analysis

A conference version of this paper appeared in the Proceedings of the 18th International Workshop on Randomization and Computation (RANDOM'14) [46].

[*]Part of this work was done while the author was at Harvard University, supported by NSF grant CCF-1116616 and the Lord Rutherford Memorial Research Fellowship.

[†]Supported in part by NSF grant CCF-1116616, US-Israel BSF grant 2010196, and a Simons Investigator Award.

[‡]Part of this work was completed while at Harvard University.

DOI: 10.4086/toc.2017.v013a012

1 Introduction

1.1 Pseudorandom generators for space-bounded computation

A central problem in the theory of pseudorandomness is constructing an "optimal" pseudorandom generator for space-bounded computation—that is, an explicit algorithm that stretches a uniformly random seed of $O(\log n)$ bits to n bits that cannot be distinguished from uniform by any $\log n$-space algorithm. Such a generator would imply that every randomized algorithm can be derandomized with only a constant-factor increase in space ($\mathsf{RL} = \mathsf{L}$), and would also have a variety of other applications, such as in streaming algorithms [26], deterministic dimension reduction and SDP rounding [43, 16], hashing [13], hardness amplification [23], almost k-wise independent permutations [27], and cryptographic pseudorandom generator constructions [21].

To construct a pseudorandom generator for space-bounded algorithms using space s, it suffices to construct a generator that is pseudorandom against ordered branching programs of width 2^s. A *read-once, oblivious branching program* B is a non-uniform model of space-bounded computation that reads one input bit at a time, maintaining a state in $[w] = \{1,\dots,w\}$, where w is called the *width* of B. At each time step $i = 1,\dots,n$, the program B can read a different input bit $x_{\pi(i)}$ (for some permutation π) and uses a different state transition function $T_i : [w] \times \{0,1\} \to [w]$. It is often useful to think of a read-once, oblivious branching program as a directed acyclic graph consisting of $n+1$ groups of w vertices each, where the i^{th} group corresponds to the *state space* at time i. The transition function defines a bipartite graph between consecutive state spaces (which we call a *layer*), where we connect state s in state space $i-1$ to states $T_i(s,0)$ and $T_i(s,1)$ in state space i (labeling those edges 0 and 1, respectively). (We usually visualise the edges as going left to right.) Figure 1 gives an example illustration of a read-once, oblivious branching program.

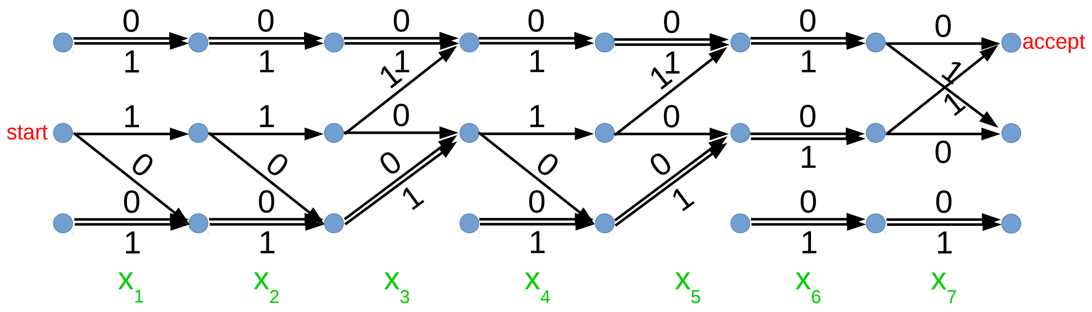

Figure 1: The graphical representation of a length-7 width-3 ordered branching program computing the function $f(x) = ((x_1 \wedge x_2 \wedge x_3) \vee (x_4 \wedge x_5)) \oplus x_7$.

Most previous constructions of pseudorandom generators for space-bounded computations consider *ordered* branching programs, where the input bits are read in order—that is, $\pi(i) = i$. The classic result of Nisan [33] gave a generator stretching $O(\log^2 n)$ uniformly random bits to n bits that are pseudorandom against ordered branching programs of polynomial width.[1] Despite intensive study, this is the best

[1]We say that a generator $G : \{0,1\}^\ell \to \{0,1\}^n$ is ε-pseudorandom against a program $P : \{0,1\}^n \to \{0,1\}$ (or ε-fools P) if

known seed length for ordered branching programs even of width 3, but a variety of results have shown improvements for restricted classes of ordered branching programs such as width-2 programs [39, 5], and "regular" or "permutation" ordered branching programs (of constant width) [9, 10, 28, 14, 45].[2] For width 3, hitting set generators (a relaxation of pseudorandom generators) have been constructed [42, 18]. The vast majority of these results are based on Nisan's original generator or its variants by Impagliazzo, Nisan, and Wigderson [25] and Nisan and Zuckerman [34], and adhere to a paradigm that seems unlikely to yield generators against general logspace computations with seed length better than $\log^{1.99} n$ (see [10]).

All known analyses of Nisan's generator and its variants rely on the order in which the output bits are fed to the ordered branching program (given by the permutation π). Tzur [48, Corollary 3.18] showed that this is inherent by constructing a permutation π and a constant-width ordered branching program that can distinguish the output of an instantiation of Nisan's generator when permuted according to π from uniformly random bits. The search for new ideas leads us to ask: *Can we construct a pseudorandom generator whose analysis does not depend on the order in which the bits are read?* Answering this question will hopefully yield new techniques and insights that shed light on the ordered case as well.

Another motivation for considering unordered, read-once, oblivious branching programs is that they also encompass natural classes of read-once formulas, like read-once CNFs and DNFs (which can be simulated in width 3), read-once polynomials over $GF(2)$ (simulable in width 4), and read-once AC^0 or even read-once ACC^0 (both simulable in constant width).[3]

A recent line of work [6, 24, 36] has constructed pseudorandom generators for unordered, read-once, oblivious branching programs (where the bits are fed to the branching program in an arbitrary, fixed order); however, none match both the seed length and generality of Nisan's result. For unordered branching programs of length n and width w, Impagliazzo, Meka, and Zuckerman [24] give seed length $O((nw)^{1/2+o(1)})$ improving on the linear seed length $(1-\Omega(1))\cdot n$ of Bogdanov, Papakonstantinou, and Wan [6].[4] Reingold, Steinke, and Vadhan [36] achieve seed length $O(w^2 \log^2 n)$ for the restricted class of *permutation* ordered branching programs, in which $T_i(\cdot,b)$ is a permutation on $[w]$ for all $i \in [n]$ and $b \in \{0,1\}$.

Recently, a new approach for constructing pseudorandom generators has been suggested in the paper of Gopalan et al. [18]; they constructed pseudorandom generators for read-once CNF formulas and combinatorial rectangles, and hitting set generators for ordered width-3 branching programs, all having

$|\mathbb{E}[P(G(U_\ell))] - \mathbb{E}[P(U_n)]| \le \varepsilon$, where U_ℓ and U_n are uniform on $\{0,1\}^\ell$ and on $\{0,1\}^n$, respectively.

[2]If the transition function $T_i(\cdot,b)$ is a permutation on $[w]$ for all $i \in [n]$ and $b \in \{0,1\}$, we say that the ordered branching program is a *permutation ordered branching program. Regular ordered branching programs* are defined by a weaker property, namely, for every $i \in [n]$ and $u \in [w]$, there are exactly two $(v,b) \in [w] \times \{0,1\}$ such that $T_i(v,b) = u$.

[3]The fact that every read-once ACC^0 circuit can be simulated by a read-once, oblivious, constant-width branching program can be proved by induction on the depth of the ACC^0 circuit. Indeed, it also applies to a generalization of ACC^0 where we replace the unbounded-fan-in mod m gates of ACC^0 with unbounded fan-in gates that compute products in any constant-size semigroup. (More precisely, such a gate takes input bits $(b_1,\ldots,b_t)$ and outputs 1 iff $g_1^{(b_1)} g_2^{(b_2)} \cdots g_t^{(b_t)} \in S$, where $g_1^{(0)}, g_1^{(1)}, g_2^{(0)}, g_2^{(1)}, \ldots, g_t^{(0)}, g_t^{(1)}$ are semigroup elements associated with the gate and S is a subset of the semigroup associated with the gate.) Conversely, any read-once, oblivious, constant-width branching program can be simulated by a single semigroup-product gate, taking the semigroup to be the set of functions from $[w]$ to $[w]$ under composition, where w is the width of the branching program, taking $g_i^{(b)}$ to be the transition function of the branching program at time step i upon reading a value b, and taking S to be the set of functions that map the start state to an accept state. Thus these two models are actually equivalent.

[4]A generator with seed length $\widetilde{O}(\sqrt{n}\log w)$ is given in [36]. The generator in [24] also extends to branching programs that read their inputs more than once and in an adaptively chosen order, which is more general than the model we consider.

seed length $\widetilde{O}(\log n)$ (even for polynomially small error). Their basic generator (e. g., for read-once CNF formulas) works by pseudorandomly partitioning the bits into several groups and assigning the bits in each group using a small-bias generator [31]. A key insight in their analysis is that the small-bias generator only needs to fool the function "on average," where the average is taken over the possible assignments to subsequent groups, which is a weaker requirement than fooling the original function or even a random restriction of the original function. (For a more precise explanation, see Section 4.1.)

The analysis of Gopalan et al. [18] does not rely on the order in which the output bits are read, and the previously mentioned results of Reingold, Steinke, and Vadhan [36] use Fourier analysis of regular, read-once, oblivious branching programs to show that the generator of Gopalan et al. fools unordered, read-once, oblivious permutation branching programs.

In this article, we further develop Fourier analysis of read-once, oblivious branching programs. Our main result shows that the pseudorandom generator of Gopalan et al. with seed length $\widetilde{O}(\log^3 n)$ fools width-3, read-once, oblivious branching programs.

Theorem 1.1 (Main result)**.** *There is an explicit pseudorandom generator*

$$G : \{0,1\}^{O(\log^3 n \cdot \log\log n)} \to \{0,1\}^n$$

against oblivious, read-once (but unordered), branching programs of width 3 *and length n.*

The previous best seed length for this model is the aforementioned length of $O(n^{1/2+o(1)})$ given in [24]. The construction of the generator in Theorem 1.1 is essentially the same as the generator of Gopalan et al. [18] for read-once CNF formulas, which was used by Reingold et al. [36] for read-once, oblivious, permutation branching programs. In our analysis, we give a new bound on the Fourier mass of oblivious, read-once, width-3 branching programs.

1.2 Fourier growth of branching programs

For a function $f : \{0,1\}^n \to \mathbb{R}$, let

$$\widehat{f}[s] = \mathop{\mathbb{E}}_{U}\left[f[U] \cdot (-1)^{s \cdot U}\right]$$

be the standard Fourier transform over $\mathbb{Z}_2^n$, where U is a random variable distributed uniformly over $\{0,1\}^n$ and $s \subseteq [n]$ or, equivalently, $s \in \{0,1\}^n$. The *Fourier mass* of f (also called the spectral norm of f), defined as $L(f) := \sum_{s \neq \emptyset} |\widehat{f}[s]|$, is a fundamental measure of complexity for Boolean functions (see, e. g., [19]), and its study has applications to learning theory [29, 30], communication complexity [20, 1, 47, 41], and circuit complexity [8, 11, 12]. In the study of pseudorandomness, it is well known that small-bias generators[5] with bias ε/L (which can be sampled using a seed of length $O(\log(n \cdot L/\varepsilon))$ [31, 2]) will ε-fool any function whose Fourier mass is at most L. Width-2, read-once, oblivious branching programs have Fourier mass at most $O(n)$ [5, 39] and are thus fooled by small-bias generators with bias ε/n. Unfortunately, such a bound does not hold even for very simple width-3 programs. For example, the "mod 3 function," which indicates when the Hamming weight of its input is a multiple of 3, has Fourier mass exponential in n.

[5]A small-bias generator with bias μ outputs a random variable $X \in \{0,1\}^n$ such that $\left|\mathbb{E}_X\left[(-1)^{s \cdot X}\right]\right| \leq \mu$ for every $s \subset [n]$ with $s \neq \emptyset$.

However, a more refined measure of Fourier mass is possible and often useful: let $L^k(f) = \sum_{|s|=k} |\widehat{f}[s]|$ be the *level-k Fourier mass* of f. A bound on the *Fourier growth* of f, or the rate at which $L^k(f)$ grows with k, was used by Mansour [30] to obtain an improved query algorithm for polynomial-size DNF; the junta approximation results of Friedgut [17] and Bourgain [7] are proved using approximating functions that have slow Fourier growth. This notion turns out to be useful in the analysis of pseudorandom generators as well: Reingold et al. [36] show that the generator of Gopalan et al. [18] will work if there is a good bound on the Fourier mass of low-order coefficients. More precisely, they show that for any class $\mathcal{C}$ of functions computed by read-once, oblivious, branching programs that is closed under restrictions and decompositions and satisfies $L^k(f) \le \mathrm{poly}(n) \cdot c^k$ for every k and $f \in \mathcal{C}$, there is a pseudorandom generator with seed length $\widetilde{O}(c \cdot \log^2 n)$ that fools every $f \in \mathcal{C}$. They then bound the Fourier growth of read-once, oblivious, permutation branching programs (and the even more general model of read-once, oblivious, "regular" branching programs, where each layer between consecutive state spaces is a regular bipartite graph) to obtain a pseudorandom generator for read-once, oblivious, permutation branching programs. We state their result.

Theorem 1.2 (Reingold et al. [36, Theorem 1.4])**.** *Let $f : \{0,1\}^n \to \{0,1\}$ be computed by a length-n, width-w, read-once, oblivious,* regular *branching program. Then, for all $k \in [n]$, we have $L^k(f) \le (2w^2)^k$.*

In particular, the mod 3 function over $O(k)$ bits, which is computed by an ordered permutation branching program of width 3, has Fourier mass $2^{\Theta(k)}$ at level k (for $k = o(n)$). However, the Tribes function,[6] which is also computed by an ordered, width-3 branching program, has Fourier mass $\Theta_k(\log^k n)$ at level k, so the bound in Theorem 1.2 does not hold for non-regular, read-once, oblivious branching programs even of width 3.

The Coin Theorem of Brody and Verbin [10] implies a related result: essentially, a function f computed by a width-w, length-n, read-once, oblivious branching program cannot distinguish product distributions on $\{0,1\}^n$ any better than a function satisfying $L^k(f) \le (\log n)^{O(wk)}$ for all k. To be more precise, if $X \in \{0,1\}^n$ is n independent samples from a coin with bias β (that is, each bit has expectation $(1+\beta)/2$), then $\mathbb{E}_X[f[X]] = \sum_s \widehat{f}[s]\beta^{|s|}$ for any f. Consequently, if $L^k(f) \le (\log n)^{O(wk)}$ for all k, then

$$\left| \mathop{\mathbb{E}}_{X}[f[X]] - \mathop{\mathbb{E}}_{U}[f[U]] \right| = \left| \sum_{s \ne 0} \widehat{f}[s]\beta^{|s|} \right| \le \sum_{k \in [n]} L^k(f)|\beta|^k \le O(|\beta|(\log n)^{O(w)}),$$

assuming $|\beta| \le 1/(\log n)^{O(w)}$. Brody and Verbin prove that, if f is computed by a length-n, width-w, read-once, oblivious branching program, then $|\mathbb{E}_X[f[X]] - \mathbb{E}_U[f[U]]| \le O(|\beta|(\log n)^{O(w)})$. This suggests the following potential strengthening of their result.

Conjecture 1.3 (Reingold et al. [36, Conjecture 8.1])**.** *For every constant w, the following holds. Let $f : \{0,1\}^n \to \{0,1\}$ be computed by a width-w, read-once, oblivious branching program. Then*

$$L^k(f) \le n^{O(1)} \cdot (\log n)^{O(k)},$$

where the constants in the $O(\cdot)$ expressions may depend on w.

[6]The Tribes function (introduced by Ben-Or and Linial [4]) is a DNF formula where all the terms are the same size and every input appears exactly once. The size of the clauses in this case is chosen to give an asymptotically constant acceptance probability on uniform input.

We now state the main contribution of this article, the proof of this conjecture for $w = 3$.

Theorem 1.4 (Fourier growth for width 3)**.** *Let $f : \{0,1\}^n \to \{0,1\}$ be computed by a width-3, read-once, oblivious branching program. Then, for all $k \in [n]$,*

$$L^k(f) := \sum_{s:|s|=k} |\widehat{f}[s]| \leq n^2 \cdot (O(\log n))^k .$$

This bound, combined with the techniques of Reingold et al. [36], implies our main result (Theorem 1.1). The Tribes function of [4] shows that the base of $O(\log n)$ of the Fourier growth in Theorem 1.4 is tight up to a factor of $\log \log n$.

We also prove Conjecture 1.3 with $k = 1$ for any constant width w. We state the result.

Theorem 1.5. *Let $f : \{0,1\}^n \to \{0,1\}$ be computed by a width-w, length-n, read-once, oblivious branching program. Then*

$$L^1(f) = \sum_{i\in[n]} |\widehat{f}[\{i\}]| \leq O(\log n)^{w-2} .$$

1.3 Techniques

The intuition behind our approach begins with two extreme cases of width-3, read-once, oblivious branching programs: regular branching programs and branching programs in which every layer is a non-regular layer.

Regular, ordered branching programs "mix" well: on a uniform random input, the distribution over states gets closer to uniform (in ℓ_2 distance) within each "connected component" of that layer. The connected components of each layer are the connected components of the *undirected* bipartite graph corresponding to that layer; Figure 2 illustrates connected components of layers. We can combine this

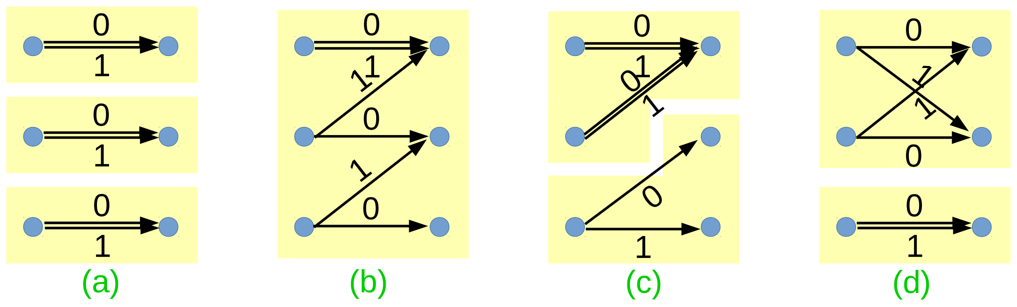

Figure 2: Examples of possible width-3 layers with connected components highlighted.

fact with an inductive argument to achieve a bound of $2^{O(k)}$ on the level-k Fourier mass. This is the bound of Theorem 1.2, which we use as part of our proof.

For ordered branching programs in which *every* layer is a non-regular layer, we can make use of an argument of Brody and Verbin [10]: when we apply a random restriction (where each variable is kept free with probability roughly $1/k \log n$) to such a branching program, the resulting program is "simple" in that many of the surviving state spaces have collapsed to width two. (This collapse is analogous to Håstad's switching lemma [22], which applies a similar argument to circuits, rather than branching programs.) The restricted branching program is now "almost" an ordered, width-2 branching program, which allows us to use arguments tailored to width-2, ordered branching programs, which are much easier to handle. In particular, we can use the same concept of mixing as used for regular, ordered branching programs.

To handle general width-3, ordered branching programs, which may contain an arbitrary mix of regular and non-regular layers, we group the layers into "chunks" containing exactly one non-regular layer each. We will then treat each chunk as if it was a single non-regular layer in the argument of Brody and Verbin. Instead of using an ordinary random restriction as Brody and Verbin do, we consider a series of carefully chosen restrictions similar to those in Steinberger's "interwoven hybrids" technique [44], which he used to give an alternative and tighter proof of the coin theorem of Brody and Verbin [10]. In Section 3.1, we use such restrictions to show that the level-k Fourier mass of an arbitrary width-3 ordered branching program can be bounded in terms of the level-k Fourier mass of a program D which has the following form: D can be split into $r \leq n$ ordered branching programs $D_1 \circ D_2 \circ \cdots \circ D_r$, where each D_i has at most $O(k)$ non-regular layers and the state space between consecutive blocks D_i has width 2.

We then generalize the arguments used for width-2, ordered branching programs to such branching programs. We can show that each chunk D_i is either mixing or has small Fourier growth, which suffices to bound the Fourier growth of D.

1.4 Organization

In Section 2 we introduce the definitions and tools we use in our proof. In Section 2.1 we formally define the kinds branching programs we consider and explain how to view them as matrix-valued functions, which is very convenient for our analysis. In Sections 2.4 and 2.5 we define the matrix-valued Fourier transform and explain how we use it.

We prove the upper bound on Fourier mass of oblivious, read-once, width-3 branching programs (i. e., Theorem 1.4) in Section 3 (Theorem 3.1). In Section 4 we construct and analyze our pseudorandom generator, which proves the main result (Theorem 1.1). The optimality of our Fourier growth bound is discussed in Section 5. We prove the bound on first-order coefficients (Theorem 1.5) in Section 6.

Finally, we discuss avenues for further research and the limitations of our techniques in Section 7.

2 Preliminaries

2.1 Branching programs as matrices

Rather than viewing ordered branching programs as graphs or as Boolean functions, we will usually view them as matrix-valued functions, as in [45, 36]. The advantage of this view is that it allows the use of linear-algebraic tools in addition to combinatorial tools. In particular, we can use linear algebra to formulate the concept of mixing along the lines of the second eigenvalue, which has been used extensively in pseudorandomness [35, 37].

We begin by recalling the standard combinatorial view of ordered branching programs and then translate it to our linear-algebraic view. Combinatorially, 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 $B[x](u)$.[7] B reads one bit of the input at a time (rather than reading x all at once) maintaining only a state in $[w] = \{1,2,\dots,w\}$ at each step. We capture this restriction that B computes by applying a transition function after reading each bit by requiring that B is composed of smaller programs as follows. The computation of a length-n, width-w ordered branching program $B : \{0,1\}^n \times [w] \to [w]$ can be broken into the composition of n length-1 programs $B_1, B_2, \dots, B_n : \{0,1\}^n \times [w] \to [w]$ such that[8]

$$B[x](u) = B_n[x_n](B_{n-1}[x_{n-1}](\cdots B_2[x_2](B_1[x_1](u))\cdots)).$$

We call B_i the transition function for the i^{th} layer.[9] Often we think of B as having a fixed *start state* u_0 and a set $S \subset [w]$ of *accept states*. Then B *accepts* $x \in \{0,1\}^n$ if $B[x](u_0) \in S$. We say that B *computes the Boolean function* $f : \{0,1\}^n \to \{0,1\}$ if $f(x) = 1$ if and only if $B[x](u_0) \in S$.

Our-linear algebraic view of a program $B : \{0,1\}^n \times [w] \to [w]$ is as a matrix-valued function $B : \{0,1\}^n \to \{0,1\}^{w \times w}$: for each $x \in \{0,1\}^n$, we let $B[x] \in \{0,1\}^{w \times w}$ be a matrix defined by

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

In our applications, the input x is randomly (or pseudorandomly) chosen. For a random variable X on $\{0,1\}^n$, we have $\mathbb{E}_X[B[X]] \in [0,1]^{w \times w}$. Then the entry in the u^{th} row and v^{th} column $\mathbb{E}_X[B[X]](u,v)$ 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,

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

Let B and B' be width-w programs of length n and n', respectively. We define the *concatenation* $B \circ B'$ of B and B' to be a width-w program of length $n + n'$ defined by

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

where $\cdot$ denotes matrix multiplication and $x \circ x' \in \{0,1\}^{n+n'}$ denotes the concatenation of the strings $x \in \{0,1\}^n$ and $x' \in \{0,1\}^{n'}$. Combinatorially, we run B and B' on separate inputs, but the final state of B becomes the start state of B'.

Then a length-n, width-w, *ordered branching program* (abbreviated *OBP*) 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 $B_i : \{0,1\} \to \{0,1\}^{w \times w}$ as the i^{th} *layer* of B. We can relate the matrix-valued function of the i^{th} layer to the i^{th} transition function by $B_i[x](u,v) = 1 \iff B_i[x](u) = v$. We can now write the entire computation in terms of matrix multiplication:

$$B[x] = B_1[x_1] \cdot B_2[x_2] \cdot \cdots \cdot B_n[x_n]$$

for all $x \in \{0,1\}^n$. We denote the *subprogram* of B from i to j by $B_{i\dots j} := B_i \circ B_{i+1} \circ \cdots \circ B_j$.

A length-n, width-w, ordered branching program is often viewed as a directed acyclic graph, with edges going left to right. The vertices are arranged into $n+1$ state spaces each containing w vertices. The edges are in layers going from one state space to the next and are specified by the transition functions. In

[7]We use the notation $B[x](u)$ rather than the more standard $B(x,u)$ to clarify what is the input x (in square brackets) and what is the state u (in round brackets).

[8]Equivalently, $B[x](u_0) = u_n$, where we recursively define $u_i = B_i[x_i](u_{i-1})$ for $i \in [n] = \{1,\dots,n\}$.

[9]$B_i[x](u) = T_i(u,x)$ in the notation of the discussion on page 2.

particular, there is an edge in layer i labelled x from vertex u in state space $i-1$ to vertex $B_i[x](u)$ in state space i.

We use the following notational conventions when referring to layers and state spaces of a length-n ordered branching program. Layers consist of *edges* and correspond to the B_i and are numbered from 1 to n. State spaces consist of *vertices* and correspond to the states between consecutive layers B_i and are numbered from 0 to n. The edges in layer i (B_i) go from vertices in state space $i-1$ (on the *left*) to vertices in state space i (on the *right*).

2.2 Classes of branching programs

An unordered, read-once, oblivious branching program is simply an ordered branching program composed with 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,

$$B[x] = B'[\pi(x)] = B'_1[x_{\pi(1)}] \cdot B'_2[x_{\pi(2)}] \cdot \cdots \cdot B'_n[x_{\pi(n)}],$$

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 $\mathbb{E}_X[B[X]]$ is *stochastic*—that is,

$$\sum_v \mathop{\mathbb{E}}_X [B[X]](u,v) = 1$$

for all u and $\mathbb{E}_X[B[X]](u,v) \geq 0$ for all u and v. A program B is called a *regular program* if the matrix $\mathbb{E}_U[B[U]]$ is *doubly stochastic*—that is, both $\mathbb{E}_U[B[U]]$ and its transpose $\mathbb{E}_U[B[U]]^*$ are stochastic. A program B is called a *permutation program* if $B[x]$ is a permutation matrix for every x or, equivalently, $\mathbb{E}_X[B[X]]$ is doubly stochastic for every distribution X. 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 read-once, oblivious, *regular branching program* is a branching program where each layer B_i is a regular program and likewise for a *permutation branching program*. We will refer to layer i as *regular* if B_i is a regular program and we say that layer i is *non-regular* otherwise.

Equivalently, a regular, read-once, oblivious branching program is one where, in the directed acyclic graph, each vertex has in-degree 2 (in addition to having out-degree 2) except those in the first state space—that is, each layer of edges is a regular graph (hence the name). A permutation, read-once, oblivious branching program has the additional constraint that the incoming edges have distinct labels.

A regular program B has the property that the uniform distribution is a stationary distribution of the Markov chain defined by the matrix $\mathbb{E}_U[B[U]]$—that is, if u is a uniformly random start vertex and U is a uniformly random input, then $B[U](u)$ is a uniformly random final state. Moreover, if B is a permutation program, the uniform distribution is stationary for $\mathbb{E}_X[B[X]]$ for *any* distribution X.

We also consider branching programs of varying width—some state spaces have more vertices than others. The width w_i of a particular state space i is the number of vertices in the i^{th} state space. The overall width of the branching program is the maximum width of any state space. This means that the layers B_i may give non-square matrices. Namely, $B_i[x] \in [0,1]^{w_{i-1} \times w_i}$ for all i and x, where w_i is the width of state space i.

2.3 Norms

We are interested in constructing a random variable X (the output of the pseudorandom generator) such that $\mathbb{E}_X[B[X]] \approx \mathbb{E}_U[B[U]]$, where U is uniform on $\{0,1\}^n$ and B is a width-3, read-once, oblivious branching program. Throughout we use U to denote the *uniform distribution*. The error of the pseudorandom generator will be measured by a norm of the matrix $\mathbb{E}_X[B[X]] - \mathbb{E}_U[B[U]]$.

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

$$\|A\|_\rho = \max_v \frac{\|vA\|_\rho}{\|v\|_\rho},$$

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

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

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

If $f : \{0,1\}^n \to \{0,1\}$ is a function computed by a width-w, length-n ordered branching program B, we can relate the error of f under pseudorandom input X versus uniformly random input U by

$$\left| \mathop{\mathbb{P}}_X[f(X)=1] - \mathop{\mathbb{P}}_U[f(U)=1] \right| \le \sqrt{w} \cdot \left\| \mathop{\mathbb{E}}_X[B[X]] - \mathop{\mathbb{E}}_U[B[U]] \right\|_2. \tag{2.1}$$

In particular, if $u_0 \in [w]$ is the start state of B and $S \subset [w]$ is the set of accept states of B, then $f(x) = \sum_{v \in S} B[x](u_0, v)$ for all $x \in \{0,1\}^n$. Furthermore, if there is a single accept state (which can be assumed by collapsing all the accept states into one), the factor of $\sqrt{w}$ can be omitted.

2.4 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, read-once, oblivious 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

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

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 $w \cdot w'$ entries of $B[x]$ 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

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

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 $A : \{0,1\}^n \to \mathbb{R}^{w\times w'}$ and $B : \{0,1\}^n \to \mathbb{R}^{w'\times w''}$ be matrix-valued functions. Let X, Y, 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: $\mathbb{E}_X [B[X]] = \sum_s \widehat{B}[s]\widehat{X}(s)$.*
- *Fourier Inversion for Matrices: $B[x] = \sum_s \widehat{B}[s]\chi_s(x)$.*
- *Fourier Inversion for Distributions: $\mathbb{P}_X [X = x] = \mathbb{E}_U \left[\widehat{X}(U)\chi_U(x)\right]$.*
- *Convolution for Distributions: If $Z = X \oplus Y$, then $\widehat{Z}(s) = \widehat{X}(s) \cdot \widehat{Y}(s)$.*
- *Parseval's Identity: $\sum_{s\in\{0,1\}^n} \left\|\widehat{B}[s]\right\|_{Fr}^2 = \mathbb{E}_U \left[\|B[U]\|_{Fr}^2\right]$.*
- *Convolution for Matrices: If, for all $x \in \{0,1\}^n$, we have $C[x] = \mathbb{E}_U [A[U] \cdot B[U \oplus x]]$, 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 by Bogdanov et al. [6]. If B is a length-n, width-w, ordered branching program, then, for all $s \in \{0,1\}^n$,

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

2.5 Fourier mass

We analyze small bias distributions as pseudorandom generators for read-once, oblivious 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

$$L(B) := \sum_{s\neq 0} \left\|\widehat{B}[s]\right\|_2 .$$

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

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

Note that $L(B) = \sum_{k\geq 1} L^k(B)$. We define

$$L^{\geq k}(B) := \sum_{k'\geq k} L^{k'}(B)\,.$$

We analogously define the quantities $L^{\leq k}(B)$, $L^{>k}(B)$, and $L^{<k}(B)$.

The following lemma asserts that 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 $s \in \{0,1\}^n$, we have $\widehat{B}[s] = \widehat{B'}[\pi(s)]$. In particular, $L(B) = L(B')$ and $L^k(B) = L^k(B')$ for all k.*

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

2.6 Small-bias distributions

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

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

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 $O(\log(n/\varepsilon))$ [31, 2].

Small-bias distributions are useful pseudorandom generators: an ε-biased random variable X is indistinguishable from uniform by any $\mathbb{F}_2$-linear function (a parity of a subset of the bits of X). That is, for any nonempty $s \subset [n]$, we have $|\mathbb{E}_X [\bigoplus_{i\in s} X_i] - 1/2| \leq \varepsilon/2$.

If $L(B)$ is small, then B is fooled by a small-bias distribution, as stated below.

Lemma 2.3. *Let B be a length-n, width-w, read-once, oblivious branching program. Let X be an ε-biased random variable on $\{0,1\}^n$. We have*

$$\left\| \mathop{\mathbb{E}}_X [B[X]] - \mathop{\mathbb{E}}_U [B[U]] \right\|_2 = \left\| \sum_{s\neq 0} \widehat{B}[s]\widehat{X}(s) \right\|_2 \leq L(B)\varepsilon \, .$$

It is known that for length-n, width-2, read-once, oblivious branching programs B we have $L(B) \leq O(n)$ [5], so they are fooled by $1/\mathrm{poly}(n)$-biased generators (which have seed length $O(\log n)$). But for a length-n, width-3, permutation, read-once, oblivious branching program B we can have $L(B) = 2^{\Theta(n)}$. For example, the program $B_{\mathrm{mod}\,3}$ that computes the Hamming weight of its input modulo 3 has exponential Fourier mass and indeed is not fooled by the $2^{-\Theta(n)}$-biased distribution that is uniform on $\{x \in \{0,1\}^n : |x| \bmod 3 = 0\}$. This means we cannot apply small-bias pseudorandom generators directly. However, small-bias distributions play a large part in our proof. The key is that, after some pre-processing (namely, applying an "averaging restriction"), we can ensure that $L(B)$ is small.

3 Fourier analysis of width-3 branching programs

In this section we prove a bound on the low-order Fourier mass of width-3, read-once, oblivious branching programs. This is key to the analysis of our pseudorandom generator.

Theorem 3.1. *Let B be a length-n, width-3, read-once, oblivious (but unordered) branching program. Then, for all $k \in [n]$,*

$$L^k(B) \leq n^2 \cdot O(\log n)^k .$$

In terms of Boolean functions, if $f : \{0,1\}^n \to \{0,1\}$ is computed by a 3OBP, then, for all $k \in [n]$, we have $L^k(f) \leq n^2 \cdot (O(\log n))^k$. This holds because there is a 3OBP B such that $f(x) = B[x](u,v)$ for some entry $(u,v) \in [3] \times [3]$, whence $L^k(f) \leq L^k(B)$. This yields Theorem 1.4.

The proof proceeds in two parts. The first part reduces the problem to one about branching programs of a special form, namely many state spaces have collapsed to width 2. The second part uses the mixing properties of such almost-width-2 programs to bound the Fourier mass.

3.1 Reduction of width by random restriction

The first part of our proof is a reduction from an arbitrary width-3, ordered branching program to one with many state spaces collapsed to width 2. Formally, our reduction can be stated as follows.

Proposition 3.2. *Let B be a length-n, width-3, ordered branching program (abbreviated* 3OBP*) with the first and last state spaces having width at most 2. Then, for all $k \leq m \leq n$, there exist length-n 3OBPs $D^1, \ldots, D^n$ such that*

1. *we have the bound*
$$L^k(B) \leq n \cdot \binom{m}{k} \sum_{\ell=0}^{n} 2^{-(\ell-1)(m-k)} L^k(D^\ell)$$
and

2. *each D^ℓ can be decomposed as*
$$D^\ell = D^\ell_1 \circ D^\ell_2 \circ \cdots \circ D^\ell_r \qquad (r \leq n),$$
where each D^ℓ_i is a 3OBP with at most $3\ell k$ non-regular layers and first and last state spaces of width at most 2.

Before continuing, we preview how Proposition 3.2 will be used. In Section 3.2, we will prove $L^k(D^\ell) \leq n \cdot O(\ell)^k$. Invoking Proposition 3.2 with $m = 2k+1$, we get

$$L^k(B) \leq n \cdot \binom{m}{k} \sum_{\ell=0}^{n} 2^{-(\ell-1)(m-k)} n \cdot O(\ell)^k \leq n^2 \cdot O(k)^k .$$

Finally, in Section 3.3, we show that we may essentially assume $k \leq O(\log n)$. Thus we get the Fourier growth bound $L^k(B) \leq n^2 \cdot O(\log n)^k$ of Theorem 3.1.

Now we turn our attention to proving Proposition 3.2. Our proof is based on those of Brody and Verbin [10] and Steinberger [44] for the Coin Theorem. In particular, we will apply a careful random restriction to B and argue that this gives a branching program of the form given by D^ℓ.

First we must define the notation for our random restrictions. For $g \subset [n]$ and $x \in \{0,1\}^n$, define the *restriction of B to g using x (denoted $B|_{\overline{g}\leftarrow x}$)* to be the branching program obtained by setting the inputs (eliminating layers) of B outside g to values from x and leaving the inputs in g free. More formally,

$$B|_{\overline{g}\leftarrow x}[y] = B[\text{Select}(g,y,x)]\,,$$

where

$$\text{Select}(g,y,x)_i = \begin{cases} y_i & \text{if } i \in g, \\ x_i & \text{if } i \notin g. \end{cases}$$

We prove Proposition 3.2 by considering a restriction $B|_{\overline{g}\leftarrow x}$ for a carefully chosen g and a random x. We show that it suffices to bound the Fourier growth of the restricted program $B|_{\overline{g}\leftarrow x}$ (Lemma 3.3) and that the restricted program is of the desired form D^ℓ with high probability (Lemma 3.4).

In order to specify the permutations g that we use for our random restrictions, we must appropriately decompose B as follows. Define a *chunk* to be a 3OBP with exactly one non-regular layer. We partition B into chunks $B = C_1 \circ \cdots \circ C_q$, where q is the number of non-regular layers in B. This partition is not necessarily unique; we arbitrarily fix one such partitioning for each 3OBP and simply refer to the i^{th} chunk C_i. Let $c_i \subset [n]$ be the layers corresponding to C_i.

Our random restrictions will be defined in terms of chunks, so that each chunk is either entirely restricted or entirely free. The choice of which chunks to restrict follows the "interwoven hybrids" technique of Steinberger [44]. The sets of variables we restrict are as follows. For $t \subset [m]$, define

$$g_t := \bigcup_{i\in t, 0\le j\le \frac{q-i}{m}} c_{i+m\cdot j} = \bigcup_{i\in[q]:(i \bmod m)\in t} c_i$$

and

$$G_t^k := \{s \subset g_t : |s| = k\}\,.$$

We refer to g_t as the t^{th} *group of indices* and G_t^k as the t^{th} *group of (order k) Fourier coefficients.*

The following lemma tells us that we may bound the level-k Fourier weight of B by considering a fixed subset $t \subset [m]$ of size k and bounding the level-k Fourier weight of the branching program that results by randomly restricting the variables outside of g_t.

Lemma 3.3. *Let $B : \{0,1\}^n \to \mathbb{R}^{w\times w}$ be an arbitrary matrix-valued function, $k \in [n]$, and $m \ge k$. Define g_t as above. Then*

$$L^k(B) \le \binom{m}{k} \cdot \max_{t\subset[m]:|t|=k} \mathop{\mathbb{E}}_U \left[L^k(B|_{\overline{g_t}\leftarrow U})\right].$$

Proof. Every Fourier coefficient $s \subset [n]$ of size k appears in G_t^k for some $t \subset [m]$. Note that some Fourier coefficients appear in more than one G_t. Thus

$$L^k(B) = \sum_{s\subset[n]:|s|=k} \left\|\widehat{B}[s]\right\|_2 \le \sum_{t:|t|=k} \sum_{s\in G_t^k} \left\|\widehat{B}[s]\right\|_2.$$

For all $s \in G_t^k$, by the definition of the Fourier transform and by the convexity of $\|\cdot\|_2$,

$$\left\|\widehat{B}[s]\right\|_2 = \left\|\mathop{\mathbb{E}}_U [B[U]\chi_s(U)]\right\|_2 = \left\|\mathop{\mathbb{E}}_{U,U'} \left[B_{\overline{g_t}\leftarrow U}[U']\chi_s(U')\right]\right\|_2 = \left\|\mathop{\mathbb{E}}_U \left[\widehat{B_{\overline{g_t}\leftarrow U}}[s]\right]\right\|_2 \le \mathop{\mathbb{E}}_U \left[\left\|\widehat{B_{\overline{g_t}\leftarrow U}}[s]\right\|_2\right].$$

Combining these inequalities yields

$$L^k(B) \le \sum_{t:|t|=k} \sum_{s \in G_t^k} \mathop{\mathbb{E}}_{U}\left[\left\|\widehat{B_{\overline{g_t} \leftarrow U}}[s]\right\|_2\right] = \sum_{t:|t|=k} \mathop{\mathbb{E}}_{U}\left[L^k(B|_{\overline{g_t} \leftarrow U})\right] \le \binom{m}{k} \max_{t \subset [m]:|t|=k} \mathop{\mathbb{E}}_{U}\left[L^k(B|_{\overline{g_t} \leftarrow U})\right]. \qquad \square$$

Given Lemma 3.3, we may now prove Proposition 3.2 by giving an upper bound on

$$\mathop{\mathbb{E}}_{U}\left[L^k(B|_{\overline{g_t} \leftarrow U})\right]$$

for any fixed $t \subset [m]$ with $|t| = k$. To do this, we prove that the random restriction $B|_{\overline{g_t} \leftarrow U}$ is with high probability a branching program of the form D^ℓ specified by Proposition 3.2. We state this formally.

Lemma 3.4. *Let B be a length-n 3OBP with width 2 in the first and last state spaces. Let $k, \ell \in [n]$ and $m \ge k$. Fix $t \subseteq [m]$ with $|t| = k$. Then with probability at least $1 - n \cdot 2^{-\ell \cdot (m-k)}$ over a random choice of $x \in \{0,1\}^n$, we can write*

$$B|_{\overline{g_t} \leftarrow x} = D_1 \circ D_2 \circ \cdots \circ D_r,$$

where $r \in [n]$ and each D_i is a 3OBP with at most $3\ell k$ non-regular layers and first and last state spaces of width at most 2.

Proof. The key step in the proof is the following claim.

Claim 3.5. *Let $E_1, \ldots, E_n$ be independent events and suppose $\mathbb{P}[E_i] \ge 1/2$ for all i. Then, for any a, with probability at least $1 - n \cdot 2^{-a}$ there are no m consecutive events $E_i, E_{i+1}, \ldots, E_{i+a-1}$ none of which occur, i. e.,*

$$\mathbb{P}\left[\exists i \in [n-a+1] \quad \overline{E_i} \wedge \overline{E_{i+1}} \wedge \cdots \wedge \overline{E_{i+a-1}}\right] \le n \cdot 2^{-a}.$$

Proof. By our assumptions on the events $\mathbb{P}\left[\overline{E_i} \wedge \overline{E_{i+1}} \wedge \cdots \wedge \overline{E_{i+a-1}}\right] \le 2^{-a}$ for all i. The claim now follows from a union bound. $\square$

First decompose B into chunks $B = C_1 \circ \cdots \circ C_q$.[10] When we perform the restriction $\overline{g_t} \leftarrow x$, some of the chunks "survive" the restriction (namely those C_i where $i \bmod m \in t$), while the rest are restricted; some of the restricted chunks will collapse to width 2, meaning only one or two of the vertices in their final state space will be reachable. We then regroup the surviving chunks as $B|_{\overline{g_t} \leftarrow x} = D_1 \circ D_2 \circ \cdots \circ D_r$ splitting every time there is a collapse to width two. We simply have to verify that with high probability no D_i will contain more than $3\ell k$ non-regular layers.

The key to the proof is that, when a chunk is restricted, it will reduce the width to two with probability at least one half. (This observation is also key to the results of Brody and Verbin [10] and Steinberger [44].) This is because of the non-regular layer contained in the chunk. Graphically, this layer must contain a right vertex with at most one incoming edge. With probability at least one half, this edge is not selected by the restriction and the vertex becomes unreachable, thereby reducing the width. In more detail, write the layers of $C_i = B_a \circ \cdots \circ B_b$ and let B_c be the non-regular layer; then $\mathbb{E}_U[B_c[U]]$ has a column whose sum is strictly less than 1. So there must exist a setting of $x_c \in \{0,1\}$ such that $B_c[x_c]$ has a column with sum

[10] If B has no non-regular layers, then the result holds trivially. So we may restrict out attention to the case where there is at least one non-regular layer.

less than 1. Since this sum is a non-negative integer, this means the column sum is zero. When x is such that $B_c[x_c]$ has a zero column, then, for any $d \in \{c, c+1, \ldots, b\}$, the matrix $B_c[x_c] \cdot B_{c+1}[x_{c+1}] \cdots B_d[x_d]$ also has a zero column by induction, since each each $B_d[x_d]$ has exactly one 1 in each row. (Graphically, there is only one edge leaving each vertex after the restriction.) Moreover, the restriction is independent across chunks.

Now we apply Claim 3.5. Each restricted chunk corresponds to one event, with the event being the width of that chunk collapsing. Consider $\ell \cdot m$ consecutive chunks of B. Of these, $\ell \cdot (m-k)$ are restricted—this is the number of consecutive events $a = \ell \cdot (m-k)$ we consider. By Claim 3.5, with high probability, at least one of these events will occur amongst every a consecutive events. That is, with high probability, every $\ell \cdot m$ consecutive chunks of B contains a chunk that is reduced to width two.

Assume that the conclusion of Claim 3.5 happens. It follows that every D_i is comprised of at most $\ell \cdot m$ chunks from B. (Otherwise we would split it into two blocks D_i.) Now we can bound the number of non-regular layers in each D_i. Each of the $\ell \cdot k$ surviving chunks contributes one non-regular layer. In addition, the restricted chunks can contribute non-regular layers, but only one between each pair of consecutive surviving chunks. Thus the number of non-regular layers is bounded by $3\ell k$. □

We may now complete the proof of Proposition 3.2.

Proof of Proposition 3.2. By Lemma 3.3, it suffices to bound, for every fixed t, the quantity

$$\mathop{\mathbb{E}}_{U}\left[L^k(B|_{\overline{g_t} \leftarrow U})\right].$$

We compute the expectation of

$$L^k(B|_{\overline{g_t} \leftarrow U})$$

using Lemma 3.4. To do so we must convert the tail bound of Lemma 3.4 into a bound on the expectation. Let $\ell_*(x)$ be the smallest ℓ such that we can write

$$B|_{\overline{g_t} \leftarrow x} = D_1^\ell \circ D_2^\ell \circ \cdots \circ D_{r_*}^\ell,$$

where each D_i^ℓ is a 3OBP with at most $3\ell k$ non-regular layers and the first and last state spaces have width at most two. Lemma 3.4 gives

$$\mathbb{P}\left[\ell_*(U) > \ell\right] \le n \cdot 2^{-\ell \cdot (m-k)}$$

for all ℓ. Also $\mathbb{P}\left[\ell_*(U) > n\right] = 0$ as there are at most n layers and hence at most n non-regular layers. Thus

$$\begin{aligned}\mathop{\mathbb{E}}_{U}\left[L^k(B|_{\overline{g_t} \leftarrow U})\right] &\le \sum_{\ell=0}^{n} \mathbb{P}\left[\ell_*(U) = \ell\right] \cdot \mathop{\mathbb{E}}_{x}\left[L^k(B|_{\overline{g_t} \leftarrow x}) \mid \ell_*(U) \le \ell\right] \\ &\le \sum_{\ell=0}^{n} n \cdot 2^{-(\ell-1)\cdot(m-k)} \cdot L^k(D^\ell),\end{aligned}$$

where

$$D^\ell = D_1^\ell \circ D_2^\ell \circ \cdots \circ D_{r_*}^\ell$$

is a branching program of length at most n that maximizes $L^k(D^\ell)$ subject to each D_i^ℓ having at most $3\ell k$ non-regular layers and having width two in the first and last state spaces of each D_i. □

3.2 Mixing in width 2

Now it remains to bound the Fourier mass of 3OBPs of the form given by Proposition 3.2.

Proposition 3.6. *Let D^ℓ be a length-n 3OBP such that*

$$D^\ell = D_1^\ell \circ D_2^\ell \circ \cdots \circ D_r^\ell,$$

where each D_i^ℓ is a 3OBP with at most ℓ non-regular layers and width 2 in the first and last state spaces. Then

$$L^k(D^\ell) \le 2n \cdot (10^4(\ell+1))^k = n \cdot O(\ell)^k$$

for all $k, \ell \ge 1$.

Before we prove Proposition 3.6, we show how it implies the Fourier bound of Theorem 3.1 for $k \le O(\log n)$.

Theorem 3.7. *Let B be a length-n, width-3, read-once, oblivious branching program with width 2 in the first and last state spaces. Then, for all $k \in [n]$,*

$$L^k(B) := \sum_{s \in \{0,1\}^n : |s| = k} |\widehat{B}[s]| \le 16n^2 \cdot \left(10^6 k\right)^k = n^2 \cdot O(k)^k.$$

The difference between Theorems 3.7 and 3.1 is that here we have $n^2 \cdot O(k)^k$, whereas we want $n^2 \cdot O(\log n)^k$. Later we will reduce the analysis of higher-order coefficients to case of $k \le O(\log n)$.

Proof. Let B be a 3OBP and assume B has width 2 in the first and last state spaces. By Propositions 3.2 and 3.6, we have, setting $m = 2k+1$,

$$\begin{aligned}
L^k(B) &\le n \cdot \binom{m}{k} \sum_{\ell \ge 0} 2^{-(\ell-1)(m-k)} L^k(D^\ell) \\
&\le n \cdot \binom{m}{k} \sum_{\ell \ge 0} 2^{-(\ell-1)(m-k)} 2n \cdot (10^4(3\ell k+1))^k \\
&\le 4n^2 \binom{2k+1}{k} \sum_{\ell \ge 0} 2^{-(\ell-1)(k+1)} (10^4(3\ell k+1))^k \\
&\le 4n^2 4^k \sum_{\ell \ge 0} 2^{-(\ell-1)} \cdot \left(10^4 \frac{3\ell k+1}{2^{\ell-1}}\right)^k \\
&\le 4n^2 4^k \left(\sum_{\ell \ge 0} 2^{-(\ell-1)}\right) \cdot \left(10^4 \cdot 4k\right)^k \\
&\le 16n^2 \cdot \left(10^4 \cdot 4 \cdot 4k\right)^k,
\end{aligned}$$

as required. □

A key notion in our proof of Proposition 3.6 is a measure of the extent to which an ordered branching program (or subprogram) mixes, and the way this affects the Fourier spectrum. To measure the mixing, we use a parameter that equals the second eigenvalue in the case of regular programs; this has been useful as a measure of mixing in other work on pseudorandomness for space-bounded computation [35, 37, 38].

For an ordered branching program $D : \{0,1\}^n \to \{0,1\}^{w \times w}$ of width w, define

$$\lambda(D) = \max_{v \in \mathbb{R}^w : \sum_i v_i = 0} \frac{\|v \cdot \mathbb{E}_U [D[U]]\|_2}{\|v\|_2} . \tag{3.1}$$

Intuitively, if v is a distribution on start states of a branching program D, then $v \cdot \mathbb{E}_U [D[U]]$ is the distribution on final states on uniformly random input. We are interested in how close $v \cdot \mathbb{E}_U [D[U]]$ is to uniform relative to v. To do this we decompose $v = u + v'$ where u is the uniform distribution and v' is orthogonal to the uniform distribution. Then

$$v \cdot \mathop{\mathbb{E}}_U [D[U]] = u \cdot \mathop{\mathbb{E}}_U [D[U]] + v' \cdot \mathop{\mathbb{E}}_U [D[U]] .$$

When D is regular, we have $u \cdot \mathbb{E}_U [D[U]] = u$—that is, u is a stationary distribution for the Markov chain defined by the program D. Regardless of whether D is regular or not,

$$\left\| v' \cdot \mathop{\mathbb{E}}_U [D[U]] \right\|_2 \leq \lambda(D) \|v\|_2 .$$

So

$$\left\| v \cdot \mathop{\mathbb{E}}_U [D[U]] - u \right\|_2 \leq \lambda(D) \|v - u\|_2 .$$

Thus, in the regular case, the quantity $\lambda(D)$ measures how rapidly D mixes the states towards uniform, measured in 2-norm.

If D is regular, we have $\lambda(D) \in [0,1]$, where 0 corresponds to perfect mixing and 1 to no mixing. If D is not regular, it is possible that $\lambda(D) > 1$. However, for width-$w = 2$—where $\mathbb{E}_U [D[U]]$ is a 2×2 matrix—it turns out that $\lambda(D) \leq 1$ even if D is non-regular. We now state this.

Lemma 3.8. *Let*

$$M = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix}$$

be a stochastic matrix. Then

$$\lambda(M) = \frac{\|(1,-1) \cdot M\|_2}{\|(1,-1)\|_2} = |1 - \alpha - \beta| \leq 1 .$$

Notice that $|(1-\alpha) - \beta| = |(1-\beta) - \alpha|$ measures how different the behaviour is between the two possible start states. Thus $\lambda(M)$ still captures some kind of mixing, even if it is not convergence to uniform. This lemma is crucial to our analysis and is the main reason our results do not extend to higher widths.[11] Mixing also can be used to bound Fourier coefficients via the following lemma.

[11]For further discussion of the limitations of our proof, see Section 7.

Lemma 3.9. *Let $D = D_1 \circ D_2 : \{0,1\}^{n_1+n_2} \to \{0,1\}^{w \times w}$ be the composition of programs D_1 and D_2 of lengths n_1 and n_2 respectively. Consider the Fourier coefficient $\widehat{D}[s \circ 0^{n_2}]$ for $s \in \{0,1\}^{n_1}$. Then*

$$\left\|\widehat{D}[s \circ 0^{n_2}]\right\|_2 \leq \left\|\widehat{D_1}[s]\right\|_2 \cdot \lambda(D_2).$$

Proof. First we show that, for any $s \neq 0^{n_1}$, the rows of $\widehat{D}_1[s]$ sum to zero. Since $D_1[x]$ is a stochastic matrix for all $x \in \{0,1\}^{n_1}$, we have $D_1[x] \cdot \vec{1} = \vec{1}$ for all $x \in \{0,1\}^{n_1}$, where $\vec{1}$ is the all ones vector. Thus, for $s \neq 0^{n_1}$,

$$\widehat{D}_1[s] \cdot \vec{1} = \mathop{\mathbb{E}}_{U}\left[D[U] \cdot \vec{1}\chi_s(U)\right] = \mathop{\mathbb{E}}_{U}\left[\vec{1}\chi_s(U)\right] = \vec{0}.$$

Thus

$$\left\|\widehat{D}[s \circ 0^{n_2}]\right\|_2 = \max_{v} \frac{\left\|v \cdot \widehat{D_1}[s] \cdot \widehat{D_2}[0^{n_2}]\right\|_2}{\|v\|_2} = \max_{v} \frac{\left\|v \cdot \widehat{D_1}[s]\right\|_2}{\|v\|_2} \frac{\left\|v \cdot \widehat{D_1}[s] \cdot \widehat{D_2}[0^{n_2}]\right\|_2}{\left\|v \cdot \widehat{D_1}[s]\right\|_2} \leq \left\|\widehat{D_1}[s]\right\|_2 \cdot \lambda(D_2),$$

as

$$\sum_i (v \cdot \widehat{D_1}[s])_i = 0$$

for all v. □

Mixing is a very powerful tool on its own. For ordered branching programs B in which every layer is mixing—that is $\lambda(B_i) \leq 1-\gamma < 1$ for all i—Lemma 3.9 can be used with an inductive argument (simpler than the proof below) to obtain a $1/\gamma^{O(k)}$ bound on the level-k Fourier mass.

We will use mixing as part of our proof. We show that any D_i with width two in the first and last state spaces with few non-regular layers (the form given by Proposition 3.2) will either mix well or have small Fourier mass after restriction.

To formalize this, we define the *p-damped Fourier mass* of a branching program B as

$$L_p(B) = \sum_{k>0} p^k L^k(B) = \sum_{s \neq 0} p^{|s|} \left\|\widehat{B}[s]\right\|_2.$$

We can translate p-damped Fourier mass back to level k Fourier mass: $L^k(B) \leq L_p(B) p^{-k}$ for all k and p. The main lemma we prove in this section is the following.

Lemma 3.10. *If D is a length-d 3OBP with at most $k \geq 1$ non-regular layers that has only two vertices in the first and last state spaces, then*

$$\lambda(D) + L_p(D) \leq 1$$

for any $p \leq 1/10^4(k+1)$.

First, we show that Lemma 3.10 implies Proposition 3.6.

Proof of Proposition 3.6. Let D^ℓ be 3OBP of length at most n such that $D^\ell = D_1^\ell \circ D_2^\ell \circ \cdots \circ D_r^\ell$, where each D_i^ℓ is a 3OBP with at most ℓ non-regular layers and width 2 in the first and last state spaces. Let $p = 1/10^4(\ell+1)$.

We inductively show that

$$L_p(D_1^\ell \circ \cdots \circ D_i^\ell) = L_p(D_{1\cdots i}^\ell) \leq 2i\,,$$

and hence $L_p(D) \leq 2r \leq 2n$, which implies the result. For $i = 0$ this is trivial. Now suppose it holds for i. By decomposition, we have

$$\begin{aligned}
&L_p(D_{1\cdots i}^\ell \circ D_{i+1}^\ell) \\
&= \sum_{(s,t)\neq 0} p^{|s|+|t|} \left\| \widehat{D_{1\cdots i}^\ell}[s] \cdot \widehat{D_{i+1}^\ell}[t] \right\|_2 && \text{(Lemma 2.1 Decomposition)} \\
&\leq \quad \sum_{s\neq 0} p^{|s|} \left\| \widehat{D_{1\cdots i}^\ell}[s] \right\|_2 \sum_{t\neq 0} p^{|t|} \left\| \widehat{D_{i+1}^\ell}[t] \right\|_2 \\
&\quad + \sum_{s\neq 0} p^{|s|} \left\| \widehat{D_{1\cdots i}^\ell}[s] \cdot \widehat{D_{i+1}^\ell}[0] \right\|_2 \\
&\quad + \left\| \widehat{D_{1\cdots i}^\ell}[0] \right\|_2 \sum_{t\neq 0} p^{|t|} \left\| \widehat{D_{i+1}^\ell}[t] \right\|_2 \\
&\leq \quad L_p(D_{1\cdots i}^\ell) \cdot L_p(D_{i+1}^\ell) && \text{(Lemma 3.9)} \\
&\quad + L_p(D_{1\cdots i}^\ell)\lambda(D_{i+1}^\ell) + \left\| \widehat{D_{1\cdots i}^\ell}[0] \right\|_2 \cdot L_p(D_{i+1}^\ell) \\
&\leq L_p(D_{1\cdots i}^\ell) \cdot 1 + \left\| \widehat{D_{1\cdots i}^\ell}[0] \right\|_2 \cdot L_p(D_{i+1}^\ell) && \text{(Lemma 3.10)} \\
&\leq 2i + 2\,,
\end{aligned}$$

since

$$\left\| \widehat{D_{1\cdots i}^\ell}[0] \right\|_2 \leq 2\,,$$

as this is the norm of a 2×2 stochastic matrix. Thus, we have that

$$L^k(D^\ell) \leq p^{-k} L_p(D^\ell) \leq 2n \cdot (10^4(\ell+1))^k\,,$$

as required □

Now we turn our attention to Lemma 3.10. We split into two cases. First, if $\lambda(D)$ is far from 1, i. e., $\lambda(D) \leq 0.99$, then we need only ensure $L_p(D) \leq 1/100$. This is the "easy case" which is a simple extension of the analysis of ordered regular branching programs of Reingold et al. [36]. Second, if $\lambda(D) = 1$, then we will see that D is trivial—i. e., $L_p(D) = 0$—and we are also done. The "hard case" is when $\lambda(D)$ is very close to 1, i. e., $0.99 \leq \lambda(D) < 1$.

3.2.1 Easy case—good mixing

We extend the analysis of Reingold et al. [36] to prove the following result.

Lemma 3.11. *Let D be a 3OBP with at most k non-regular layers. If $p \leq 1/10^4(k+1)$, then $L_p(D) \leq 1/100$.*

It immediately follows that, assuming $\lambda(D) < 0.99$ (i. e., good mixing), we have $\lambda(D) + L_p(D) \leq 1$ when $p \leq 1/10^4(k+1)$. This covers the "easy" case of Lemma 3.10.

We begin with the following lemma that forms the basis of the analysis of Reingold et al.

Lemma 3.12 (Braverman et al. [9, Lemma 4], reformulated by Reingold et al. [36, Lemma 3.1])**.** *Let B be a length-n, width-w, ordered, regular branching program. Then*

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

where $B_{i \cdots n}$ denotes the ordered branching program composed of layers $i, i+1, \ldots, n$ from B.

The entry in the uth row and vth column of

$$2\widehat{B_{i \cdots n}}[1 \circ 0^{n-i}] = (B_i[0] - B_i[1]) \cdot \mathbb{E}[B_{i+1 \cdots n}[U]]$$

is the the probability of reaching vertex v in state space n given that we reached vertex u in state space $i-1$ and the ith input bit is 0 minus the same probability given that the ith input bit is 1. Thus the quantity

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

measures the correlation between the ith input bit and the final state of the program, which we call the *weight* of bit i. Braverman et al. [9] proved Lemma 3.12 for a slightly different measure of weight. Their result was translated into the above Fourier-analytic form by Reingold et al. [36].

We can add some non-regular layers to get the following.

Lemma 3.13. *Let B be a length-n, width-w, ordered branching program with at most k non-regular layers. Then*

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

Proof. The proof proceeds by induction on k. If $k = 0$, the result follows from Lemma 3.12. Suppose the result holds for some k and let B be a length-n, width-w ordered branching program with $k+1$ non-regular layers. Let i^* be the index of the first non-regular layer. Then

$$\begin{aligned}\sum_{1\le i\le n}\left\|\widehat{B_{i\cdots n}}[1\circ 0^{n-i}]\right\|_2 = & \sum_{1\le i<i^*}\left\|\widehat{B_{i\cdots(i^*-1)}}[1\circ 0^{i^*-i-1}]\cdot\widehat{B_{i^*\cdots n}}[0]\right\|_2 \\ & + \left\|\widehat{B_{i^*\cdots n}}[1\circ 0^{n-i^*}]\right\|_2 \\ & + \sum_{i^*<i\le n}\left\|\widehat{B_{i\cdots n}}[1\circ 0^{n-i}]\right\|_2 \\ \le & \left(\sum_{1\le i<i^*}\left\|\widehat{B_{i\cdots(i^*-1)}}[1\circ 0^{i^*-i-1}]\right\|_2\right)\cdot\left\|\widehat{B_{i^*\cdots n}}[0]\right\|_2 \\ & + \sqrt{w}+(2w^2+1)\sqrt{w}(k+1) \\ \le & \, 2w^2\cdot\sqrt{w}+\sqrt{w}+(2w^2+1)\sqrt{w}(k+1) \\ \le & \, (2w^2+1)\sqrt{w}(k+2)\,,\end{aligned}$$

where we use the fact that $\left\|\widehat{B}[s]\right\|_2 \le \sqrt{w}$ for any s and width-w, ordered branching program B. □

This gives us the following bound on the Fourier mass.

Lemma 3.14. *Let B be a length-n, width-w, ordered branching program with at most k non-regular layers. Then, for all $k' \in [n]$,*

$$L^{k'}(B) := \sum_{s\in\{0,1\}^n:|s|=k}\left\|\widehat{B}[s]\right\|_2 \le \sqrt{w}\cdot((2w^2+1)\sqrt{w}(k+1))^{k'} \le \sqrt{w}\cdot(3w^{2.5}(k+1))^{k'}\,.$$

This result is proved using Lemma 3.13 analogously to how Theorem 1.2 was proved using Lemma 3.12 by Reingold et al. [36, Theorem 3.2].

Proof. 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 $\mathbb{E}_U[B[U]]$ is stochastic,

$$\left\|\mathop{\mathbb{E}}_{U}[B[U]]\right\|_2 \le \sqrt{w}$$

and the base case follows. Now suppose the bound holds for k' and consider $k'+1$. In the following calculation, we split the Fourier coefficients based on where the last 1 is.

$$\begin{aligned}
&\sum_{s\in\{0,1\}^n:|s|=k'+1} \left\|\widehat{B}[s]\right\|_2 \\
&\quad= \sum_{1\le i\le n}\sum_{s\in\{0,1\}^{i-1}:|s|=k'} \left\|\widehat{B}[s\circ 1\circ 0^{n-i}]\right\|_2 \\
&\quad= \sum_{1\le i\le n}\sum_{s\in\{0,1\}^{i-1}:|s|=k'} \left\|\widehat{B_{1\cdots i-1}}[s]\cdot \widehat{B_{i\cdots n}}[1\circ 0^{n-i}]\right\|_2 && \text{(by Lemma 2.1 (Decomposition))}\\
&\quad\le \sum_{1\le i\le n}\sum_{s\in\{0,1\}^{i-1}:|s|=k'} \left\|\widehat{B_{1\cdots i-1}}[s]\right\|_2\cdot \left\|\widehat{B_{i\cdots n}}[1\circ 0^{n-i}]\right\|_2 \\
&\quad\le \sum_{1\le i\le n} \sqrt{w}\cdot((2w^2+1)\sqrt{w}(k+1))^{k'}\cdot\left\|\widehat{B_{i\cdots n}}[1\circ 0^{n-i}]\right\|_2 && \text{(by the induction hypothesis)}\\
&\quad\le \sqrt{w}\cdot((2w^2+1)\sqrt{w}(k+1))^{k'}\cdot(2w^2+1)\sqrt{w}(k+1) && \text{(by Lemma 3.13)}\\
&\quad= \sqrt{w}\cdot((2w^2+1)\sqrt{w}(k+1))^{k'+1},
\end{aligned}$$

as required. □

Proof of Lemma 3.11. We have

$$\begin{aligned}
L_p(D) = \sum_{k'\ge 1} p^{k'}L^{k'}(D) &\le \sum_{k'\ge 1} p^{k'}\sqrt{3}((2\cdot 3^2+1)\sqrt{3}(k+1))^{k'}\\
&\le \sqrt{3}\sum_{k'\ge 1}\left(\frac{19\sqrt{3}(k+1)}{10^4(k+1)}\right)^{k'} \le 1/100\,.
\end{aligned}$$
□

3.2.2 Hard case—poor mixing

Now we consider the case where $\lambda(D)\in[0.99,1]$.

Lemma 3.15. *Let D be a 3OBP with at most k non-regular layers where the first and last state spaces of vertices have width 2. Suppose $\lambda(D)\in[0.99,1]$. If $p\le 1/(352k+252)$, then $L_p(D)+\lambda(D)\le 1$.*

This covers the 'hard' case of Lemma 3.10 and, along with Lemma 3.11 completes the proof of Lemma 3.10. Since D has width 2 in the first and last state spaces, we view $D[x]$ as a 2×2 matrix. Now write

$$D[x] = \begin{pmatrix} 1-f(x) & f(x)\\ g(x) & 1-g(x)\end{pmatrix},$$

where $f,g:\{0,1\}^d\to\{0,1\}$. We can write the expectation (which is stochastic) as

$$\mathop{\mathbb{E}}_U[D[U]] = \begin{pmatrix} 1-\alpha & \alpha\\ \beta & 1-\beta\end{pmatrix},$$

where $\alpha = \mathbb{E}_U[f(U)]$ and $\beta = \mathbb{E}_U[g(U)]$. By Lemma 3.8, $\lambda(D) = |1-\alpha-\beta| \in [0.99,1]$. We can assume (by permuting rows and columns) that $\alpha, \beta \in [0, 1/100]$.

We will show that $L_p(f) \leq O(kp\alpha)$ and $L_p(g) \leq O(kp\beta)$ for $p \leq 1/O(k)$. To prove Lemma 3.15, we choose $p = 1/O(k)$ such that $L_p(f) \leq \alpha/2$ and $L_p(g) \leq \beta/2$. Then, since

$$L_p(D) = \sum_{s \neq 0} p^{|s|} \left\| \begin{pmatrix} -\widehat{f}(s) & \widehat{f}(s) \\ \widehat{g}(s) & -\widehat{g}(s) \end{pmatrix} \right\|_2 \leq 2L_p(f) + 2L_p(g) \leq \alpha + \beta$$

and $\lambda(D) = 1-\alpha-\beta$, the result follows.

First, we give some intuition. We can view D as having two corresponding start and end states. The probability that, starting in the first start state, we end in the first end state is $1-\alpha \geq 0.99$. Likewise, the probability that, starting in the second start state, we end in the second end state is $1-\beta \geq 0.99$. The function f is computed by starting in the first start state and accepting if we end in the second end state—that is, we "cross over." Likewise, g computes the function telling us whether we will cross over from the second start state to the first end state. Intuitively, there is a low probability ($\leq 1/100$) of "crossing over," so the program behaves like two disjoint programs. We will formalize this intuition by showing that D can be partitioned into states corresponding to the first start+end states and states corresponding to the second start+end states. This means that the width-3, read-once, oblivious branching program can be decomposed in terms of width-2, read-once, oblivious branching programs. Then we can apply tools tailored to width-2, read-once, oblivious branching programs to analyze this decomposition.

The plan is as follows.

1. Show that we can partition the states/vertices of D with $O(k)$ edges crossing the partition such that each state space has at least one vertex on each side of the partition. Intuitively, this partitions D into two width-2, ordered branching programs with a few edges going between them.

2. Using this partition, we decompose D into width-2, ordered branching programs. We are able to show that f can be expressed as a sum (or OR) of ANDs of regular, width-2, ordered branching programs. That is, we can express $f(x) = \sum_s \prod_j f_{s,j}(x_j)$, where each $f_{s,j}$ is a $\{0,1\}$-valued function computed by a *regular*, width-2, ordered branching program, the product is over $O(k)$ terms, and the x_j form a partition of the input x.

3. Consider one term
$$f_s(x) = \prod_j f_{s,j}(x_j) \quad \text{and} \quad \alpha_s = \mathop{\mathbb{E}}_U[f_s(U)]\,.$$
Using an analysis heavily tailored to ANDs of regular, width-2, ordered branching programs, we can show that $L_p(f_s) \leq O(kp\alpha_s)$ for $p \leq 1/O(k)$. Then, by the triangle inequality,
$$L_p(f) \leq \sum_s L_p(f_s) \leq \sum_s O(kp\alpha_s) \leq O(kp\alpha)\,,$$
as required.

The same holds for g, which gives the result.

Step 1. Partitioning the states of a poorly mixing 3OBP with few non-regular layers

Lemma 3.16. *Let D be a 3OBP with at most k non-regular layers and width-2 in the first and last state spaces. Suppose $\lambda(D) \in [0.99, 1]$. Then there is a partition of the vertices of D such that each state space has at least one vertex in each side of the partition and there are at most $43k+31$ edges that cross the partition.*

Proof. We assign each vertex of D a numerical "charge". Let $v_0 = (1, -1)$. For $i \in [d]$, let $v_i = v_{i-1} \cdot \mathbb{E}_U [D_i[U]]$. The charge of vertex $u \in [w]$ in state space $i \in \{0, \ldots, d\}$ is $v_i(u)$.

In other words, the charge of a vertex (i,u) is the probability that a random execution of D starting from the first start vertex reaches (i,u) minus the probability that an execution from the second start vertex reaches (i,u).

Claim 3.17. *Every state space has a vertex with charge > 0.49 and a vertex with charge < -0.49.*

Proof. The sum of absolute values of charges cannot increase from one state space to the next, as $\mathbb{E}_U [D_i[U]]$ is stochastic. That is, $\|v_i\|_1 \le \|v_{i-1}\|_1$ for all i. Moreover,

$$\|v_d\|_1 = \left\| (1,-1) \cdot \mathop{\mathbb{E}}_U [D[U]] \right\|_1 = \left\| (1,-1) \cdot \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix} \right\|_1 = 2\lambda(D) \ge 2 \cdot 0.99 .$$

We also know that the charges sum to zero in every state space, as they are "conserved." In particular,

$$\sum_u v_i(u) = v_i \cdot \vec{1} = v_{i-1} \cdot \mathop{\mathbb{E}}_U \left[D_i[U] \cdot \vec{1} \right] = v_{i-1} \cdot \vec{1} = \sum_u v_{i-1}(u) .$$

Thus, in each state space, we have three charges whose sum is zero and whose sum of absolute values is at least $2 \cdot 0.99$. A simple case analysis shows that this implies that at least one must be larger than 0.49 and at least one must be smaller than -0.49. $\square$

Define the *length* of an edge in the ordered branching program to be the absolute difference between the charges of its endpoints. More precisely, letting (i,u) denote the vertex corresponding to the state u in state space i, the length of the edge $(i-1,u) \to (i,u')$ is $|v_{i-1}(u) - v_i(u')|$. We will analyze the sum of the lengths of the edges. To bound the sum of the edge lengths, we use the following.

Following the proof of Lemma 3.12 from [9], define the "potential" of a multiset of real numbers S as

$$\rho(S) = \sum_{a,b \in S} |a-b| .$$

Let S_i be the multiset that contains each point $v_i(u)$ twice. We are interested in the potentials $\rho(S_0), \ldots, \rho(S_d)$ of the state spaces.

Claim 3.18. *For all $i \in [n]$, if D_i is a regular layer, then*

$$\sum_{(i-1,u) \to (i,u')} \left| v_{i-1}(u) - v_i(u') \right| \le \rho(S_{i-1}) - \rho(S_i) .$$

Proof. We will change S_{i-1} to S_i two points at a time. At each step we argue that the potential decreases proportional to the length of two edges. Summing these changes will prove the claim.

Each change is of the following form. For a vertex u in state space i, we look at the two incoming edges $(i-1,u_1) \to (i,u)$ and $(i-1,u_2) \to (i,u)$ and we change one copy of $v_{i-1}(u_1)$ and one copy of $v_{i-1}(u_2)$ to two copies of $v_i(u)$. Since the layer is regular, doing this for every u will change S_{i-1} to S_i.

Now we must analyze what happens to the potential when doing this. Let $S_{\text{before } u}$ and $S_{\text{after } u}$ denote the set before and after changing the two points corresponding to u.

We have

$$\rho(S_{\text{before } u}) - \rho(S_{\text{after } u}) = |v_{i-1}(u_1) - v_{i-1}(u_2)| + \sum_a |v_{i-1}(u_1) - a| + |v_{i-1}(u_2) - a| - 2|v_i(u) - a|\,.$$

Since $v_i(u) = (v_{i-1}(u_1) + v_{i-1}(u_2))/2$, by the triangle inequality

$$|v_{i-1}(u_1) - a| + |v_{i-1}(u_2) - a| - 2|v_i(u) - a| \geq 0$$

for all a. Thus

$$\rho(S_{\text{before } u}) - \rho(S_{\text{after } u}) \geq |v_{i-1}(u_1) - v_{i-1}(u_2)| = |v_{i-1}(u_1) - v_i(u)| + |v_{i-1}(u_2) - v_i(u)|\,.$$

The right hand side is simply the sum of the lengths of the two edges entering (i,u). Now, summing over all u proves the claim.

$$\rho(S_{i-1}) - \rho(S_i) = \sum_u \rho(S_{\text{before } u}) - \rho(S_{\text{after } u}) \geq \sum_{(i-1,u)\to(i,u')} \left|v_{i-1}(u) - v_i(u')\right|\,. \qquad \square$$

Now we have a claim bounding the sum of edge lengths.

Claim 3.19.

$$\sum_{i\in[d]} \sum_{(i-1,u)\to(i,u')} \left|v_{i-1}(u) - v_i(u')\right| \leq 12k + 30(k+1)\,.$$

Proof. Firstly, $0 \leq \rho(S_i) \leq 30$ for all i, as $\|v_i\|_\infty \leq 1$ and $|S_i| \leq 6$. If D_i is a regular layer, then

$$\sum_{(i-1,u)\to(i,u')} \left|v_{i-1}(u) - v_i(u')\right| \leq \rho(S_{i-1}) - \rho(S_i)\,.$$

We use this to bound the contribution from regular layers. If there are only regular layers, we would be able to say that the sum of edge lengths is bounded by 30, but we must account for the non-regular layers.

For a non-regular layer D_i, we have

$$\sum_{(i-1,u)\to(i,u')} \left|v_{i-1}(u) - v_i(u')\right| \leq 12\,,$$

as there are 6 edges each of length at most 2. A non-regular layer may also increase the potential by 30. Each non-regular layer will contribute at most 12 to the sum of edge weights with its own edges and 30 by raising the potential function for the regular layers. $\square$

Now we can define our partition. For $t \in [-0.49, 0.49]$ define a partition $(Q_t, \overline{Q_t})$ by whether the charge of each vertex is above or below the threshold t. Namely,

$$Q_t = \{(i,u) \in \{0,\ldots,n\} \times [w] : v_i(u) \geq t\}.$$

An edge $(i-1,u) \to (i,u')$ crosses the partition $(Q_t, \overline{Q_t})$ if and only if

$$v_{i-1}(u) \geq t > v_i(u') \quad \text{or} \quad v_{i-1}(u) < t \leq v_i(u').$$

If $t \in [-0.49, 0.49]$ is chosen uniformly at random, the probability that an edge $(i-1,u) \to (i,u')$ crosses the partition is at most

$$\frac{1}{0.49+0.49}|v_{i-1}(u) - v_i(u')|,$$

slightly more than its length.

By linearity of expectations, the expected number of edges crossing the partition for a random $t \in [-0.49, 0.49]$ is at most

$$\frac{1}{0.98}\sum_{i\in[d]}\sum_{(i-1,u)\to(i,u')}\left|v_{i-1}(u) - v_i(u')\right| \leq \frac{42k+30}{0.98}.$$

In particular, there exists some fixed choice of $t \in [-0.49, 0.49]$ such that at most $43k+31$ edges cross the partition $(Q_t, \overline{Q_t})$.

By the first claim, each state space has a vertex with charge greater than t and a vertex with charge less than t. Thus this partition has at least one vertex on each side in every state space. □

Step 2. Decomposing a 3OBP into a sum of ANDs of regular, width-2, ordered branching programs.

Lemma 3.20. *Let D be a length-d 3OBP with at most k non-regular layers and width-2 in the first and last state spaces. Suppose $\lambda(D) \geq 0.99$. Let $f : \{0,1\}^n \to \{0,1\}$ be a function computed by D, namely $f(x) = D[x](1,1)$. Then we can write $f(x) = \sum_s \prod_j f_{s,j}(x_j)$, where each $f_{s,j}$ is computed by a regular, width-2, ordered branching program and the x_j's are a partition of x into at most $88k+63$ contiguous parts.*

Proof. Fix a partition of the vertices of D as promised by Lemma 3.16. Call a layer D_i *critical* if it is either non-regular or it has an edge crossing the partition. Let $\Gamma \subset [d]$ denote the set of critical layers. By Lemma 3.16, D has at most $44k+31$ critical layers.

Between critical layers, D is partitioned into two width-2, regular, ordered branching programs. That is, if D_i and D_j are consecutive critical layers then $D_{i+1\cdots j-1}$ is a regular ordered branching program and, viewed as an undirected graph, it is disconnected; the connected components of $D_{i+1\cdots j-1}$ are two ordered branching programs of width at most 2 (or three width-1 ordered branching programs).

The value of f is entirely determined by what happens in the critical layers. In particular, it is determined by (the parity of) the number of times the partition is crossed. To make use of this observation, we consider all possibilities for what happens in the critical layers.

Formally, we define $\widetilde{\Gamma}$ to be the set of possibilities for critical edges—that is, $s \in \widetilde{\Gamma}$ specifies for each $i \in \Gamma$ an edge $s(i)$ in layer i (specifying one of three states to the left of layer i and the label, which is 0 or 1, of the edge taken). We can think of $s \in \widetilde{\Gamma}$ as a function $s : \Gamma \to [3] \times \{0,1\}$. Let $\widetilde{\Gamma}_1$ denote the set of possibilities that imply $f(x) = 1$.

Given a possibility $s \in \widetilde{\Gamma}$ for what happens in the critical layers, we want to be able to compute whether or not that possibility happened. That is, for $s \in \widetilde{\Gamma}$, we want to compute the indicator function $f_s : \{0,1\}^d \to \{0,1\}$ given by

$$f_s(x) = 1 \iff \text{the path in } D \text{ determined by input } x \text{ uses all the edges specified by } s\,.$$

Now

$$f(x) = \sum_{s \in \widetilde{\Gamma}_1} f_s(x)\,,$$

as each path in D is consistent with exactly one s. So if we can compute f_s, we can compute f.

We claim that each f_s can be written as the conjunction of at most $2|\Gamma| + 1$ functions computed by regular, width-2, ordered branching programs. In particular, there is one term for each critical layer and one term for each gap between critical layers. Once we prove this claim, the lemma follows.

Let $i_1 < i_2 < \cdots < i_{|\Gamma|}$ be an enumeration of Γ. (Also define $i_0 = 0$ and $i_{|\Gamma|+1} = d+1$.) We will write

$$f_s(x) = \prod_{j=1}^{2|\Gamma|+1} f_{s,j}(x_j)\,,$$

where the x_j form a partition of x as follows. For $j \in [|\Gamma|]$, let $x_{2j} \in \{0,1\}$ be coordinate i_j of x. For $j \in [|\Gamma|+1]$, let $x_{2j-1} \in \{0,1\}^{i_j - i_{j-1} - 1}$ contain the coordinates of x from $i_{j-1}+1$ to $i_j - 1$. The function $f_{s,j}$ checks the bits in x_j and is 1 if and only if the path is consistent with $f_s = 1$, assuming the bits outside of x_j are set consistently with $f_s = 1$. This is computable by a regular, width-2, ordered branching program. For a critical layer, $f_{s,2j}(x_{2j})$ is simply x_{2j} or $\neg x_{2j}$. Between critical layers, D is partitioned into two regular, width-2, ordered branching programs and $f_{s,2j-1}$ must simply simulate one of these. □

Step 3. Tailored analysis of a sum of ANDs of regular, width-2, ordered branching programs We use the following fact about regular width-2 ordered branching programs—a very simple class of functions, that permits a case analysis.

Lemma 3.21. *Let $f : \{0,1\}^n \to \{0,1\}$ be computed by a width-2 regular ordered branching program. Then $\mathbb{E}_U[f(U)] \in \{0, 1/2, 1\}$ and $L_p(f) \le p/2$ for all $p \in [0,1]$.*

Proof. Every layer of a regular width-2 ordered branching program falls into one of three cases: trivial layers (the input bit does not affect the state), a (negated) XOR (flips the state depending on the input bit), or a (negated) dictator (sets the state based on the current input bit regardless of the previous state). Thus any such branching program is either a constant function, which gives $\mathbb{E}_U[f(U)] \in \{0,1\}$ and $L_p(f) = 0$, or is a (possibly negated) XOR of a subset of the input bits. In the latter case f has one non-trivial Fourier coefficient of magnitude $1/2$, which implies $\mathbb{E}_U[f(U)] = 1/2$ and $L_p(f) \le p/2$. □

Lemma 3.22. *Let $f : \{0,1\}^n \to \{0,1\}$ be of the form $f(x) = \prod_{j\in[k]} f_j(x_j)$, where the x_j form a partition of x and each f_j is computed by a width-2, ordered, regular branching program. Then $L_p(f) \le 2kp \cdot \mathbb{E}_U[f(U)]$ for $p \le 1/k$.*

Proof. Define $\alpha_j = \mathbb{E}_U[f_j(U)]$. We have $\alpha = \prod_{j\in[k]} \alpha_j$. Now

$$\alpha + L_p(f) = \sum_s p^{|s|}|\widehat{f}[s]| = \sum_s p^{|s|} \prod_{j\in[k]} |\widehat{f_j}[s_j]| = \prod_{j\in[k]} \sum_{s_j} p^{|s_j|}|\widehat{f_j}[s_j]| = \prod_{j\in[k]} (\alpha_j + L_p(f_j)).$$

Since each f_j is computed by a width-2, regular, ordered branching program, $\alpha_j \in \{0, 1/2, 1\}$ by Lemma 3.21. If $\alpha_j = 0$, then $f_j = 0$, whence $f = 0$ and the result is trivially true; so this case is covered and we do not consider it further. Moreover, if $\alpha_j = 1$, then $f_j = 1$ and $L_p(f_j) = 0$. Thus we can ignore the factors with $\alpha_j = 1$, as they are just 1. Let $J = \{j \in [k] : \alpha_j = 1/2\}$. Thus we are left with

$$\begin{aligned} L_p(f) &= \prod_{j\in J}\left(\frac{1}{2} + L_p(f_j)\right) - 2^{-|J|} = \alpha \cdot \left(\prod_{j\in J}(1 + 2L_p(f_j)) - 1\right) \le \alpha \cdot \left(\prod_{j\in J}(1+p) - 1\right) \\ &\le \alpha \cdot \left((\mathrm{e}^p)^{|J|} - 1\right) \le \alpha \cdot 2p|J| \le \alpha 2pk, \end{aligned}$$

as long as $p|J| \le 1$. □

Now we complete the analysis of the poor mixing case.

Proof of Lemma 3.15. Let D be a 3OBP with at most k non-regular layers, where the first and last state spaces have width two and $\lambda(D) \in [0.99, 1]$. Fix $p \le 1/4(88k+63)$. Write

$$D[x] = \begin{pmatrix} 1 - f(x) & f(x) \\ g(x) & 1 - g(x) \end{pmatrix}$$

and

$$\mathbb{E}_U[D[U]] = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix},$$

where $f, g : \{0,1\}^d \to \{0,1\}$ are functions and $\alpha = \mathbb{E}_U[f(U)]$ and $\beta = \mathbb{E}_U[g(U)]$ are their expectations. By Lemma 3.8, $\lambda(D) = |1 - \alpha - \beta| \in [0.99, 1]$. We can assume that $\alpha, \beta \in [0, 1/100]$.

By Lemma 3.20, we can write $f(x) = \sum_s \prod_j f_{s,j}(x_j)$, where the product is over at most $88k + 63$ terms, the x_j form a partition of x, and each $f_{s,j}$ is computed by a width-2, regular, ordered branching program. Let $f_s(x) = \prod_j f_{s,j}(x_j)$. Then, by Lemma 3.22,

$$L_p(f) \le \sum_s L_p(f_s) \le \sum_s 2(88k+63)p \cdot \mathbb{E}_U[f_s(U)] = 2(88k+63)p\alpha \le \alpha/2.$$

Likewise $L_p(g) \le \beta/2$.

Now

$$L_p(D) = \sum_{s\neq 0} p^{|s|} \left\| \begin{pmatrix} 1 - f(x) & f(x) \\ g(x) & 1 - g(x) \end{pmatrix} \right\|_2 \le 2L_p(f) + 2L_p(g) \le \alpha + \beta$$

and

$$L_p(D) + \lambda(D) \le (\alpha + \beta) + (1 - \alpha - \beta) \le 1,$$

as required. □

3.3 Bootstrapping

Theorem 3.7 gives a bound on the Fourier growth of width-3, read-once, oblivious branching programs of the form $L^k(B) \leq n^2 \cdot O(k)^k$. The k^k term is inconvenient, but can be easily removed by "bootstrapping." The following proposition shows that if we can bound the Fourier mass up to level $k^* = O(\log n)$, then we can bound the Fourier mass at all levels k.

Proposition 3.23. *Fix parameters k^*, a, b, n with $a \cdot n \leq 2^{k^*}$. Let B be a length-n, ordered branching program such that, for all $i, j, k \in [n]$ with $k \leq 2k^*$ and $i \leq j$, we have $L^k(B_{i\cdots j}) \leq a \cdot b^k$, where $B_{i\cdots j}$ denotes the branching program consisting of layers i through j of B. Then, for all $i, j, k \in [n]$ with $i \leq j$, we have $L^k(B_{i\cdots j}) \leq a \cdot (2b)^k$.*

Proof. Suppose the proposition is false and fix the smallest $k \geq 0$ such that $L^k(B_{i\cdots j}) > a \cdot (2b)^k$ for some $i \leq j$. By assumption, if $k \leq 2k^*$, then $L^k(B_{i\cdots j}) \leq a \cdot b^k$. Thus $k > 2k^*$. Let $k' = k - k^*$. By the minimality of our choice of k and the fact that $k' < k$, we have $L^{k'}(B_{i\cdots j}) \leq a \cdot (2b)^{k'}$ for all $i \leq j$. Now

$$\begin{aligned}
L^k(B_{i\cdots j}) &= \sum_{s \subset \{1,\ldots,j-i+1\},\ |s|=k} \left\| \widehat{B}_{i\cdots j}[s] \right\|_2 \\
&\leq \sum_{\ell=i}^{j+1} \sum_{\substack{s_1 \subset \{1,\ldots,\ell-i\},\quad |s_1| = k', \\ s_2 \subset \{\ell-i+1,\ldots,j-i+1\},\quad |s_2| = k^*}} \left\| \widehat{B}_{i\cdots j}[s_1 \circ s_2] \right\|_2 \\
&= \sum_{\ell=i}^{j+1} \sum_{s_1 \subset \{1,\ldots,\ell-i\},\ |s_1|=k'} \sum_{s_2 \subset \{\ell-i+1,\ldots,j-i+1\},\ |s_2|=k^*} \left\| \widehat{B}_{i\cdots \ell-1}[s_1] \cdot \widehat{B}_{\ell\cdots j}[s_2] \right\|_2 \\
&\leq \sum_{\ell=i}^{j+1} \sum_{s_1 \subset \{1,\ldots,\ell-i\},\ |s_1|=k'} \sum_{s_2 \subset \{\ell-i+1,\ldots,j-i+1\},\ |s_2|=k^*} \left\| \widehat{B}_{i\cdots \ell-1}[s_1] \right\|_2 \cdot \left\| \widehat{B}_{\ell\cdots j}[s_2] \right\|_2 \\
&= \sum_{\ell=i}^{j+1} L^{k'}(B_{i\cdots \ell-1}) \cdot L^{k^*}(B_{\ell\cdots j}) \\
&\leq \sum_{\ell=i}^{j+1} a \cdot (2b)^{k'} \cdot a \cdot b^{k^*} \\
&\leq n \cdot a \cdot (2b)^{k'} \cdot a \cdot b^{k^*} \\
&= a \cdot (2b)^k \cdot \left(\frac{na}{2^{k^*}} \right).
\end{aligned}$$

Since $na \leq 2^{k^*}$, we have a contradiction, as we assumed $L^k(B_{i\cdots j}) > a \cdot (2b)^k$. □

Now we combine Theorem 3.7 with Proposition 3.23 to prove Theorem 3.1.

Proof of Theorem 3.1. Let B be a length-n 3OBP with width 2 in the first and last state spaces. (The assumption of having width 2 in the first and last state spaces can be removed at the expense of a constant factor increase in the bound, as each entry in the matrix $B[x]$ can be equated with an entry in the matrix of a 3OBP with width 2 in the first and last state spaces by simply collapsing the irrelevant entries.) By Theorem 3.7, we have

$$L^k(B_{i\cdots j}) \le 16n^2 \cdot \left(10^6 k\right)^k$$

for all $i, j, k \in [n]$. Now we set the "bootstrapping level" $k^* = \lceil \log_2(16n^3) \rceil = \Theta(\log n)$ and the Fourier growth parameters $a = 16n^2$ and $b = 10^6 \cdot 2k^*$ to satisfy the hypotheses of Proposition 3.23. Thus, for any $k \in [n]$, we have

$$L^k(B) \le 16n^2 \cdot \left(2 \cdot 10^6 \cdot 2k^*\right)^k ,$$

which implies the theorem. □

4 The pseudorandom generator

Our main result Theorem 1.1 follows from plugging our Fourier growth bound (Theorem 3.1) into the analysis of Reingold et al. [36]. Unfortunately, Reingold et al. do not have a general theorem linking Fourier growth to a pseudorandom generator, although this is implicit. For completeness, we include a general statement here and give a proof.

Theorem 4.1. *Let $\mathcal{C}$ be a set of ordered branching programs of length at most n and width at most w that is closed under restrictions and subprograms—that is, if $B \in \mathcal{C}$, then $B|_{\bar{t} \leftarrow x} \in \mathcal{C}$ for all t and x and $B_{i\cdots j} \in \mathcal{C}$ for all i and j. Suppose that, for all $B \in \mathcal{C}$ and $k \in [n]$, we have $L^k(B) \le ab^k$, where $b \ge 2$. Let $\varepsilon > 0$.*

Then there exists a pseudorandom generator

$$G_{a,b,n,w,\varepsilon} : \{0,1\}^{s_{a,b,n,w,\varepsilon}} \to \{0,1\}^n$$

with seed length

$$s_{a,b,n,w,\varepsilon} = O\left(b \cdot \log(b) \cdot \log(n) \cdot \log\left(\frac{abwn}{\varepsilon}\right)\right)$$

such that, for any length-n, width-w, read-once, oblivious (but unordered) branching program B that corresponds to an ordered branching program in $\mathcal{C}$,[12]

$$\left\| \underset{U_{s_{a,b,n,w,\varepsilon}}}{\mathbb{E}} \left[B[G_{a,b,n,\varepsilon}(U_{s_{a,b,n,w,\varepsilon}})]\right] - \underset{U}{\mathbb{E}}[B[U]] \right\|_2 \le \varepsilon .$$

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

To prove Theorem 1.1 we set $\mathcal{C}$ to be the class of all 3OBPs of length at most n. Theorem 3.1 gives a bound corresponding to $a = O(n^2)$ and $b = O(\log n)$. This gives the required generator. The statements of Theorems 1.1 and 4.1 differ in that Theorem 4.1 bounds the error of the pseudorandom generator with respect to a matrix-valued function, while Theorem 1.1 bounds the error with respect to a $\{0,1\}$-valued function. These statements are equivalent as the $\{0,1\}$-valued function is simply one entry in the matrix-valued function; see (2.1).

The generator is the same as that used by Reingold et al. [36] and Gopalan et al. [18].

[12]That is, there exists $B' \in \mathcal{C}$ and a permutation of the bits $\pi : \{0,1\}^n \to \{0,1\}^n$ such that $B[x] = B'[\pi(x)]$ for all x.

4.1 Pseudorandom restrictions

Our pseudorandom generator repeatedly applies pseudorandom restrictions. For the analysis, we introduce the concept of an averaging restriction as in Gopalan et al. [18] and Reingold et al. [36], which is subtly different from the restrictions in Section 3.

Definition 4.2. For $t \in \{0,1\}^n$ and a length-n branching program B, let $B|_t$ be the *(averaging) 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

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

where U is uniform on $\{0,1\}^n$.

In this section we show that, for a *pseudorandom* T (generated using few random bits), if B has good Fourier growth, then $L(B|_T)$ is small with high probability. In particular, we generate T using an almost $O(\log n)$-wise independent distribution, which we now define.

Definition 4.3. A random variable X on Ω^n is *δ-almost k-wise independent* if, for every index set $I = \{i_1, i_2, \ldots, i_k\} \subset [n]$ with $|I| = k$, the coordinates $(X_{i_1}, X_{i_2}, \ldots, X_{i_k}) \in \Omega^k$ are δ-close (in statistical distance) to being independent—that is, for all $T \subset \Omega^k$,

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

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 [36, Lemma B.2].

The following lemma, proved in essentially the same way as Lemma 5.3 in [36], tells us that $L(B|_T)$ will be small for T chosen from a δ-almost k-wise distribution with appropriate parameters.

Lemma 4.4. *Let B be a length-n, width-w, ordered 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 that, for all $i, j, k' \in [n]$ such that $k \le k' < 2k$, we have $L^{k'}(B_{i\cdots j}) \le a\cdot b^{k'}$. If we set $p \le 1/2b$ and $\delta \le 1/(2b)^{2k}$, then*

$$\mathop{\mathbb{P}}_{T}\left[L^{\ge k}(B|_T) > 1\right] \le n^4 \cdot \frac{2a}{2^k}\,.$$

(Recall that $L^{\ge k}(g) = \sum_{j=k}^{n} L^j(g)$.)

The proof is similar to that of Proposition 3.23.

Proof. Let $k \le k' < 2k$. We have that for all i and j,

$$\mathop{\mathbb{E}}_{T}\left[L^{k'}(B_{i\cdots j}|_T)\right] = \sum_{s\subset\{i\cdots j\}:|s|=k'}\mathop{\mathbb{P}}_{T}[s\subset T]\left\|\widehat{B_{i\cdots j}[s]}\right\|_2 \le L^{k'}(B)(p^{k'}+\delta) \le ab^{k'}\left(\frac{1}{(2b)^{k'}}+\frac{1}{(2b)^{2k}}\right) \le \frac{2a}{2^k}\,.$$

Applying Markov's inequality and a union bound, we have that, for all $\beta > 0$,

$$\mathop{\mathbb{P}}_{T}\left[\forall 1 \le i \le j \le n \quad L^{k'}(B_{i\cdots j}|_T) \le \beta\right] \ge 1 - n^2 \frac{2a}{2^k \beta}.$$

Applying a union bound over values of k' and setting $\beta = 1/n$, we obtain

$$\mathop{\mathbb{P}}_{T}\left[\forall k \le k' < 2k \, \forall 1 \le i \le j \le n \quad L^{k'}(B_{i\cdots j}|_T) \le \frac{1}{n}\right] \ge 1 - n^4 \cdot \frac{2a}{2^k}.$$

The result now follows from the following Lemma.

Lemma 4.5 (Reingold et al. [36, Lemma 5.4])**.** *Let B be a length-n, ordered branching program and $t \in \{0,1\}^n$. Suppose that, for all integers i, j, k' with $1 \le i \le j \le n$ and $k \le k' < 2k$, we have $L^{k'}(B_{i\cdots j}|_t) \le 1/n$. Then, for all $k'' \ge k$ and all i and j, we have $L^{k''}(B_{i\cdots j}|_t) \le 1/n$.*

The proof of Lemma 4.5 is similar to that of Proposition 3.23. □

4.2 Pseudorandom generator construction

The following lemma gives the basis of our pseudorandom generator. In our application, the Fourier growth parameters are $a = O(n^2)$ and $b = O(\log n)$ and the independence parameters are $\delta = 1/n^{O(\log\log n)}$ and $k = O(\log n)$.

Lemma 4.6. *Let the parameters a and b and the class $\mathcal{C}$ be as in Theorem 4.1 and $B \in \mathcal{C}$. 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*

$$p \le 1/(2b), \qquad k \ge \log_2\left(8an^4 w/\varepsilon\right), \quad \text{and} \quad \delta \le 1/(2b)^{2k}.$$

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

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

Thus $\mathrm{Select}(T,X,U)$ is a pseudorandom distribution. To complete the construction we will recursively generate $\widetilde{U}$ and use $\mathrm{Select}(T,X,\widetilde{U})$ as the pseudorandom output.

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

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

Conditioning on whether or not $L^{\geq k}(B|_T) > 1$, we have

$$\begin{aligned}
&\left\| \mathop{\mathbb{E}}_{T,X,U}[B[\mathrm{Select}(T,X,U)]] - \mathop{\mathbb{E}}_{U}[B[U]] \right\|_2 \\
&\qquad \leq \mathop{\mathbb{P}}_{T}\left[L^{\geq k}(B|_T) > 1\right] \max_t \left\| \mathop{\mathbb{E}}_{X,U}[B[\mathrm{Select}(t,X,U)]] - \mathop{\mathbb{E}}_{U}[B[U]] \right\|_2 \\
&\qquad\quad + \mathop{\mathbb{P}}_{T}\left[L^{\geq k}(B|_T) \leq 1\right] \mu \mathop{\mathbb{E}}_{T}\left[L(B|_T) \;\middle|\; L^{\geq k}(B|_T) \leq 1\right].
\end{aligned}$$

We have

$$L^{<k}(B) \leq \sum_{1 \leq k' < k} ab^{k'} = ab\frac{b^{k-1}-1}{b-1} \leq ab^k - 1\,.$$

Thus $\mathbb{E}_T\left[L(B|_T) \,\middle|\, L^{\geq k}(B|_T) \leq 1\right] \leq ab^k$. Lemma 4.4 gives

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

For all t,x,y, we have $\|B[\mathrm{Select}(t,x,y)] - \mathbb{E}_U[B[U]]\|_2 \leq 2w$. Thus

$$\begin{aligned}
\left\| \mathop{\mathbb{E}}_{T,X,U}[B[\mathrm{Select}(T,X,U)]] - \mathop{\mathbb{E}}_{U}[B[U]] \right\|_2 &\leq n^4 \cdot \frac{2a}{2^k} \cdot 2w + 1 \cdot \mu \cdot ab^k \\
&\leq \frac{4an^4w}{8an^4w/\varepsilon} + ab^k \frac{\varepsilon}{2ab^k} \\
&\leq \varepsilon\,.
\end{aligned}$$

□

Now we use the above results to construct our pseudorandom generator.
The pseudorandom generator is formally defined as follows.

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

```
Parameters: $n \in \mathbb{N}$ and $\varepsilon > 0$.
     Input: A random seed of length $s_{a,b,n,\varepsilon}$.
        1. Compute appropriate values of parameters satisfying[13]

           $p \leq 1/2b$,   $\varepsilon' = \varepsilon p/14w\log_2(n)$,   $k \geq \log_2(8an^4w/\varepsilon')$,
           $\delta \leq \varepsilon'(p/2)^{2k}$,   $\mu \leq \varepsilon'/2ab^k$.

           However, if the algorithm is being called recursively, use the parameter values from the
           previous level of recursion.[14]
        2. If $n \leq 320 \cdot \lceil \log_2(1/\varepsilon') \rceil / p$, output $n$ truly random bits and stop.
```

[13]Pick parameter values that are within a constant factor of these constraints.

[14]For the purposes of the analysis we assume that $\varepsilon', k, p, \delta, \mu$ are the same at every level of recursion.

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 $\widetilde{U} \in \{0,1\}^{\lfloor n(1-p/2) \rfloor}$, i. e., $\widetilde{U} = G_{a,b,\lfloor n(1-p/2) \rfloor,\varepsilon}(U)$.
6. Sample $X \in \{0,1\}^n$ from a μ-biased distribution.
7. Output $\mathrm{Select}(T,X,\widetilde{U}) \in \{0,1\}^n$.[15]

The analysis of the algorithm proceeds roughly as follows.

- We have

$$p = \Theta(1/b)\,, \qquad \varepsilon' = \Theta(\varepsilon/wb\log n)\,, \qquad k = \Theta(\log(abwn/\varepsilon))\,,$$
$$\delta = 1/b^{\Theta(k)}\,, \qquad \mu = 1/b^{\Theta(k)}\,.$$

- 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 $r = O(\log(n)/p) = O(b\log n)$.

- 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

$$\mathop{\mathbb{E}}_{U}[B[G_{a,b,n,\varepsilon}(U)]] = \mathop{\mathbb{E}}_{T,X,\widetilde{U}}\left[B[\mathrm{Select}(T,X,\widetilde{U})]\right] \approx \mathop{\mathbb{E}}_{T,X,U}[B[\mathrm{Select}(T,X,U)]] \approx \mathop{\mathbb{E}}_{U}[B[U]]\,.$$

  The first approximate equality holds because we inductively assume that $\widetilde{U}$ is pseudorandom. The second approximate equality holds by Lemma 4.6.

- The total seed length is the seed length needed to sample X and T at each level of recursion and $O(\log(1/\varepsilon')/p) = O(b\log(bwn/\varepsilon))$ truly random bits at the last level. Sampling X requires seed length $O(\log(n/\mu)) = O(k\log b)$ and sampling T requires seed length

$$O(k\log(1/p) + \log(1/\delta) + \log(1/p)\cdot\log(n\log(1/p))) = O(k\log b)$$

  so the total seed length is

$$r\cdot O(k\log b) + O(b\log(bwn/\varepsilon)) = O\left(b\cdot\log(b)\cdot\log(n)\cdot\log\left(\frac{abwn}{\varepsilon}\right)\right).$$

Lemma 4.7. *The probability that $G_{a,b,n,\varepsilon}$ fails at step 4 is bounded by $3\varepsilon'$—that is, $\mathbb{P}_T\left[|T| < pn/2\right] \le 3\varepsilon'$.*

Proof. We use the following extension of a standard result [40, Theorem 4].

[15]Technically, we must pad $\widetilde{U}$ with zeros in the locations specified by T (i. e., $\widetilde{U}_i = 0$ for $i \in T$) to obtain the right length.

Lemma 4.8 (Tail bound for limited independence)**.** *Let $X_1,\ldots,X_\ell$ be δ-almost k-wise independent random variables with $X_i \in \{0,1\}$ for all i. Set $X = \sum_i X_i$ and $\mu = \sum_i \mu_i = \sum_i \mathbb{E}_X[X_i]$, and suppose $\mu_i \le 1/2$ for all i. If $k \le \mu/10$ is even, then, for all $\alpha \in (0,1)$,*

$$\mathop{\mathbb{P}}_X[|X-\mu| \ge \alpha\mu] \le \left(\frac{20k}{\alpha^2\mu}\right)^{\lfloor k/2 \rfloor} + 2\delta \cdot \left(\frac{\ell}{\alpha\mu}\right)^k .$$

Proof of Lemma 4.8. Assume, without loss of generality, that k is even. It is well known [40, 32, 3] that, if the X_i are fully independent, then

$$\mathop{\mathbb{E}}_{X_i}\left[(X-\mu)^k\right] \le (20k\mu)^{k/2} .$$

This also holds when the X_i are only k-wise independent, as $(X-\mu)^k$ is a degree-k polynomial in the X_i. Here the X_i are δ-almost k-wise independent, which gives

$$\mathop{\mathbb{E}}_{X_i}\left[(X-\mu)^k\right] \le (20k\mu)^{k/2} + 2\delta\ell^k .$$

Thus we can apply Markov's inequality to obtain the result, as follows.

$$\mathop{\mathbb{P}}_X[|X-\mu| \ge \alpha\mu] = \mathop{\mathbb{P}}_X\left[(X-\mu)^k \ge (\alpha\mu)^k\right] \le \frac{\mathbb{E}_{X_i}\left[(X-\mu)^k\right]}{(\alpha\mu)^k} \le \left(\frac{20k\mu}{\alpha^2\mu^2}\right)^{k/2} + 2\delta\left(\frac{\ell}{\alpha\mu}\right)^k . \qquad \square$$

By the Chernoff bound for limited independence (Lemma 4.8),

$$\mathop{\mathbb{P}}_T[|T| < pn/2] \le \left(\frac{20k'}{(1/2)^2 pn}\right)^{\lfloor k'/2 \rfloor} + 2\delta \cdot \left(\frac{n}{(1/2)pn}\right)^{k'} ,$$

where $k' \le 2k$ even is arbitrary. Set $k' = 2\lceil \log_2(1/\varepsilon') \rceil$. Step 2 ensures that $n > 160k'/p$ and our setting of δ gives that $\delta \le \varepsilon'(p/2)^{k'}$. Thus we have

$$\mathop{\mathbb{P}}_T[|T| < pn/2] \le 2^{-\log_2(1/\varepsilon')} + 2\varepsilon' \le 3\varepsilon' . \qquad \square$$

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

Lemma 4.9. *Let $B \in \mathcal{C}$. Then*

$$\left\| \mathop{\mathbb{E}}_{U_{s_{n,\varepsilon}}}\left[B[G_{n,\varepsilon}(U_{s_{n,\varepsilon}})]\right] - \mathop{\mathbb{E}}_U[B[U]] \right\|_2 \le 7wr\varepsilon' ,$$

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

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

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

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

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

By Lemma 4.6,

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

By Lemma 4.7,

$$\mathop{\mathbb{P}}_{T} [|T| < pn/2] \le 3\varepsilon' \,.$$

We claim that

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

Before we prove the claim, we complete the proof. This gives $\Delta_i \le \Delta_{i+1} + \varepsilon' + 2w \cdot 3\varepsilon'$. It follows that $\Delta_0 \le 7wr\varepsilon'$, as required.

To prove the claim, consider *any* fixed $T_i = t$ and $X_i = x$. We have

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

Consider $\overline{B}_{x,t}[y] := B'[\mathrm{Select}(t, x, y)]$ as a function of $y \in \{0,1\}^{n_i - |t|}$. Then $\overline{B}_{x,t}$ is a width-3, read-once, oblivious branching program of length-$(n_i - |t|)$.

We inductively know that $\widetilde{U}_i$ is pseudorandom for $\overline{B}_{x,t}$—that is,

$$\left\| \mathop{\mathbb{E}}_{\widetilde{U}_i} \left[\overline{B}_{x,t}[\widetilde{U}_i] \right] - \mathop{\mathbb{E}}_{U} \left[\overline{B}_{x,t}[U] \right] \right\|_2 \le \Delta_{i+1} \,.$$

Thus

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

as required. □

Proof of Theorem 4.1. Since $\varepsilon' \leq \varepsilon/(7wr)$, Lemma 4.9 implies that $G_{a,b,n,\varepsilon}$ has error at most ε. The seed length is

$$s_{a,b,n,\varepsilon} = O\left(b \cdot \log(b) \cdot \log(n) \cdot \log\left(\frac{abwn}{\varepsilon}\right)\right)$$

as required. $\square$

5 Optimality of the Fourier growth bound

The following result shows that Theorem 3.1 is close to optimal.

Proposition 5.1. *There exists an infinite family of functions $f_n : \{0,1\}^n \to \{0,1\}$ and that are computed by 3OBPs such that the following holds. Let $q : \mathbb{N} \to \mathbb{R}$ be an increasing function with $20 \leq q(n) \leq \exp(\exp(o(\sqrt{\log n})))$. For all sufficiently large n, there exists $k \in [n]$ such that*

$$L^k(f_n) > q(n) \cdot \left(\frac{\log n}{20 \log\log q(n)}\right)^k .$$

Our main result shows that $L^k(f_n) \leq \mathrm{poly}(n) \cdot (O(\log n))^k$. Setting $q(n) = \mathrm{poly}(n)$, this proposition shows that the base $O(\log n)$ cannot be improved by more than a $\log\log n$ factor. The $\log\log n$ factor comes from the fact that we allow a polynomial factor $q(n)$ in the bound. If we demand $q(n) = O(1)$, the base $O(\log n)$ is optimal.

Proof. Let $n = m \cdot 2^m$, where m is an integer. Define $f_n : \{0,1\}^n \to \{0,1\}$ by

$$f_n(x) = \prod_{i \in [2^m]} \left(1 - \prod_{j \in [m]} x_{i,j}\right),$$

where we view $x \in \{0,1\}^n$ as a $2^m \times m$ matrix $x \in \{0,1\}^{2^m \times m}$. This is (up to a negation) the Tribes function [4]. This function can be computed by a 3OBP. Now we show that it has large Fourier growth.

The Fourier coefficients of f_n are given as follows. For $s \subset [n]$ (which we identify with $s \in \{0,1\}^{2^m \times m}$),

$$\begin{aligned}
\widehat{f_n}[s] &= \mathop{\mathbb{E}}_{U}[f(U)\chi_s(U)] \\
&= \mathop{\mathbb{E}}_{U}\left[\prod_{i \in [2^m]}\left(1 - \prod_{j \in [m]} U_{i,j}\right)\chi_{s_i}(U_i)\right] \\
&= \prod_{i \in [2^m]} \mathop{\mathbb{E}}_{U}\left[\chi_{s_i}(U_i) - \prod_{j \in [m]} U_{i,j}\chi_{s_i}(U_i)\right] \\
&= \prod_{i \in [2^m]} \left(\mathbb{I}(s_i = 0) - 2^{-m}(-1)^{|s_i|}\right),
\end{aligned}$$

where $s_i = (s_{i,1}, s_{i,2}, \ldots, s_{i,m})$. The damped Fourier mass is also easy to compute. For $p \in (0,1)$,

$$\begin{aligned}
L_p(f_n) + |\widehat{f_n}[0]| &= \sum_{s \in \{0,1\}^{2^m \times m}} p^{|s|} \left|\widehat{f_n}[s]\right| \\
&= \sum_{s \in \{0,1\}^{2^m \times m}} \prod_{i \in [2^m]} p^{|s_i|} \left|\mathbb{I}(s_i = 0) - 2^{-m}(-1)^{|s_i|}\right| \\
&= \prod_{i \in [2^m]} \sum_{s_i \in \{0,1\}^m} p^{|s_i|} \left|\mathbb{I}(s_i = 0) - 2^{-m}(-1)^{|s_i|}\right| \\
&= \prod_{i \in [2^m]} \left(1 - 2^{-m} + \sum_{s_i \neq 0} p^{|s_i|} 2^{-m}\right) \\
&= \prod_{i \in [2^m]} \left(1 - 2^{-m} + 2^{-m}(1+p)^m - 2^{-m}\right) \\
&= \left(1 + \frac{(1+p)^m - 2}{2^m}\right)^{2^m}.
\end{aligned}$$

Set $p = (1 + \log(3 + \log q(n)))/m$. We have

$$(1+p)^m = \left(1 + \frac{1 + \log(3 + \log q(n))}{m}\right)^m = \mathrm{e}^{1 + \log(3 + \log q(n))}(1 - o(1)) \geq 3 + \log q(n)$$

for sufficiently large m. Thus, for sufficiently large m,

$$L_p(f_n) \geq \left(1 + \frac{(1+p)^m - 2}{2^m}\right)^{2^m} - 1 \geq \left(1 + \frac{1 + \log q(n)}{2^m}\right)^{2^m} - 1 = \mathrm{e}^{1 + \log q(n)}(1 - o(1)) - 1 \geq q(n)\,.$$

Suppose for the sake of contradiction that $L^k(f_n) \leq q(n) \cdot (\log n / 20 \log\log q(n))^k$ for all $k \in [n]$. We have

$$\begin{aligned}
L_p(f_n) = \sum_{k \in [n]} p^k L^k(f_n) &\leq \sum_{k \in [n]} \left(\frac{1 + \log(3 + \log q(n))}{m}\right)^k \cdot q(n) \cdot \left(\frac{\log n}{20 \log\log q(n)}\right)^k \\
&\leq \sum_{k \in [n]} q(n) 2^{-k} < q(n)\,,
\end{aligned}$$

which is a contradiction. □

A more careful analysis gives the following bound.

Proposition 5.2. *There exists an infinite family of functions $f_n : \{0,1\}^n \to \{0,1\}$ that are computed by 3OBPs such that, for all $k \in [n]$,*

$$L^k(f) \geq \Omega\left(\frac{\log n}{\log k}\right)^k.$$

Proof. Let n, m, and f_n be as in the proof of Proposition 5.1. For $s \in \{0,1\}^{2^m \times m}$, denote

$$\ell(s) = |\{i \in [2^m] : s_i \neq 0\}| = |\{i \in [2^m] : \exists j \in [m] \quad s_{i,j} = 1\}|\,.$$

Then, for all $s \in \{0,1\}^{n \times m}$, we have

$$|\widehat{f_n}[s]| = \prod_{i \in [2^m]} \left| \mathbb{I}(s_i = 0) - 2^{-m}(-1)^{|s_i|} \right| = (1-2^{-m})^{2^m - \ell(s)} \cdot (2^{-m})^{\ell(s)}\,.$$

Fix ℓ with $k/\ell \le m$. Set $h = \lfloor k/\ell \rfloor$. Choose $i, j \ge 0$ with $i + j = \ell$ and $ih + j(h+1) = k$. Then

$$\begin{aligned}
L^k(f_n) &\ge \sum_{|s|=k \wedge \ell(s)=\ell} |\widehat{f_n}[s]| \\
&\ge \binom{2^m}{\ell}\binom{m}{h}^i \binom{m}{h+1}^j \cdot (1-2^{-m})^{2^m-\ell} \cdot (2^{-m})^\ell \\
&\ge \left(\frac{2^m}{\ell}\right)^\ell \left(\frac{m}{h}\right)^{hi} \left(\frac{m}{h+1}\right)^{(h+1)j} \cdot \left(1 - \frac{1}{2^m}\right)^{2^m} \cdot \left(\frac{1}{2^m}\right)^\ell \\
&\ge \frac{1}{4}\left(\frac{1}{\ell}\right)^\ell \left(\frac{m}{h}\right)^{hi} \left(\frac{m}{h+1}\right)^{(h+1)j} \\
&\ge \frac{1}{4}\left(\frac{1}{\ell}\right)^\ell \left(\frac{m}{h+1}\right)^{hi+(h+1)j} \\
&\ge \frac{1}{4}\left(\frac{1}{\ell}\right)^\ell \cdot \left(\frac{m}{k/\ell+1}\right)^k .
\end{aligned}$$

Suppose $k \le 2^{m-1}$. Setting $\ell = \lceil k/\log_2(2k) \rceil$, we have

$$L^k(f_n) \ge \frac{1}{4}\left(\frac{1}{\ell}\right)^\ell \left(\frac{m}{k/\ell+1}\right)^k \ge \frac{1}{4} \cdot \frac{1}{2^{2k+2}} \cdot \left(\frac{m}{\log_2 k + 2}\right)^k ,$$

as

$$\log_2(\ell^\ell) = \ell \log_2 \ell < \frac{k + \log_2 k}{\log_2 k} \log_2(k + \log_2 k) \le 2k+2\,.$$

Since $m = \Theta(\log n)$, this gives the result for $k \le 2^{m-1}$. If $k > 2^{m-1}$, then $\log k = \Theta(\log n)$ and the result is trivial. □

6 First-order Fourier coefficients of read-once, oblivious branching programs

In addition to proving a Fourier growth bound for read-once, oblivious branching programs of width 3, our techniques give a bound on the first-order Fourier mass of all constant-width, read-once, oblivious branching programs, which we now state.

Theorem 6.1. *Let B be a width-w, length-n, read-once, oblivious branching program. Then*

$$\sum_{i\in[n]} \left\|\widehat{B}[\{i\}]\right\|_2 \le O(\log n)^{w-2}.$$

The proof is similar to the proof of the Coin Theorem by Steinberger [44]. The main difference is that we need a variant of the "collision lemma" of Brody and Verbin [10]. We call a layer B_i of an ordered branching program *trivial* if $B_i[0] = B_i[1]$ and nontrivial otherwise. We say that a layer B_i has a *collision* if there exist two edges with the same label and the same endpoint, but different start points. A layer has a collision if and only if it is not a permutation layer. If there is a collision in layer i, then with probability at least $1/2$ a random restriction of layer i reduces the width—this is what drives Lemma 3.4.

Lemma 6.2 (Collision lemma)**.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a function computed by a width-w, ordered branching program B. Then there exists a function g computed by a width-w, ordered branching program B' such that every nontrivial layer of B' has a collision and*

$$\sum_i |\widehat{f}[\{i\}]| \le \sum_i |\widehat{g}[\{i\}]|. \tag{6.1}$$

Lemma 6.2 says that, for any function f computed by an ordered branching program, we can assume that the branching program has a collision in every nontrivial layer without decreasing the level-one Fourier mass. In contrast, the original collision lemma of Brody and Verbin [10] says that, for any f computed by an ordered branching program, we can assume that every nontrivial layer has a collision without decreasing its ability to distinguish two coins of differing bias. That is, (6.1) is replaced with $\mathbb{P}[f(X)=1]-\mathbb{P}[f(Y)=1] \le \mathbb{P}[g(X)=1]-\mathbb{P}[g(Y)=1]$, where X is the distribution on $\{0,1\}^n$ given by n independent flips of a coin with bias $1/2+\varepsilon$ and Y is n flips of a coin with bias $1/2-\varepsilon$.

Proof. We will construct B' by flipping edge labels in B (viewed as a graph). Begin by ranking the vertices in each state space by their acceptance probability—that is, the probability that $f(U)=1$ conditioned on passing through that vertex. (If two vertices have the same acceptance probability rank them arbitrarily.) We will flip the edge labels in each layer such that for every vertex the 0-labelled edge leads to a higher-ranked vertex than the 1-labelled edge (or they lead to the same vertex).

Note that flipping the edges in layer i only affects the corresponding Fourier coefficient. Thus we need only show that flipping the edges in layer i does not decrease $|\widehat{f}[i]|$.

Fix a layer i. For a vertex u on the left of layer i of B (i. e., in state space $i-1$), let u_0 and u_1 be the vertices led to by the 0- and 1-edges respectively and let p_u be the probability that a random execution of B reaches u. For a vertex v on the right of layer i of B (i. e., in state space i), let q_v be the acceptance probability of B under uniform input conditioned on passing through vertex v. Then

$$\widehat{f}[\{i\}] = \frac{1}{2}\sum_u p_u(q_{u_0} - q_{u_1}).$$

Flipping edge labels corresponds to flipping the signs of the terms in the above sum as we swap q_{u_0} and q_{u_1}. Clearly, $|\widehat{f}[\{i\}]|$ is maximized if every term has the same sign. Our choice of flips ensures this is the case, as $q_{u_0} \ge q_{u_1}$.

It remains to verify that every nontrivial layer of B' must have a collision. Nonregular layers have a collision (regardless of the edge labels). Thus we need only analyze regular layers. Consider a regular layer i and let v be the highest ranked vertex on its right such that the incoming edges are from different vertices on the left. Suppose, for the sake of contradiction, that the incoming edges have different labels. Pick the edge labelled 1 and let u be its start point. Let u_0 be the vertex reached from u by the edge labelled 0. Then, by our choice of labels u_0 is ranked higher than v—a contradiction, as v is the highest ranked vertex with incoming edges with different start points. (The other incoming edge of u_0 can't also come from u, as u has an edge going to v.) □

Define $\xi(n,w)$ to be the maximal first-order Fourier mass of any function computed by a length-n, width-w, ordered branching program. We follow the structure of Steinberger's proof [44] and inductively bound ξ.

Lemma 6.3. *For all n and $w \geq 3$, we have $\xi(n,w) \leq \lceil 1+2\log_2(n)\rceil \cdot (\xi(n,w-1)+1)$.*

Proof. Let B be a length-n, width-w, ordered branching program that maximizes the first-order Fourier mass of the function f it computes. By Lemma 6.2, we may assume that every nontrivial layer of B has a collision. We may assume that there are no trivial layers. (Otherwise we can remove them without affecting the Fourier mass.)

Let $m = \lceil 1+2\log_2 n\rceil$. Split the first-order Fourier coefficients into m groups of the form

$$G_{i'} = \{i \in [n] : i \bmod m = i'\}.$$

We bound the first-order Fourier mass of each group separately and add them up, i. e.,

$$\sum_{i\in[n]} \left|\widehat{f}[\{i\}]\right| = \sum_{i'\in[m]} \sum_{i\in G_{i'}} \left|\widehat{f}[\{i\}]\right|.$$

Fix one group $G = G_{i'}$.

We apply a random restriction to B to obtain the function $f|_{\overline{G}\leftarrow U}$ computed by the branching program $B|_{\overline{G}\leftarrow U}$. As in Lemma 3.3, we have

$$\sum_{i\in G} \left|\widehat{f}[\{i\}]\right| \leq \mathop{\mathbb{E}}_{U}\left[\sum_{i\in G}\left|\widehat{f|_{\overline{G}\leftarrow U}}[\{i\}]\right|\right].$$

So if suffices to bound the first-order Fourier mass of $f|_{\overline{G}\leftarrow U}$.

We claim that $B|_{\overline{G}\leftarrow U}$ is a width-$(w-1)$, ordered branching program with probability at least $1-n\cdot 2^{1-m}$ over the choice of $U \in \{0,1\}^n$. This implies that

$$\mathop{\mathbb{E}}_{U}\left[\sum_{i\in G}\left|\widehat{f|_{\overline{G}\leftarrow U}}[\{i\}]\right|\right] \leq \xi(n,w-1) + n\cdot 2^{1-m}\cdot n \leq \xi(n,w-1)+1.$$

Thus

$$\sum_{i\in[n]} \left|\widehat{f}[\{i\}]\right| \leq \sum_{i'\in[m]} \mathop{\mathbb{E}}_{U}\left[\sum_{i\in G_{i'}}\left|\widehat{f|_{\overline{G_{i'}}\leftarrow U}}[\{i\}]\right|\right] \leq \sum_{i'\in[m]} \xi(n,w-1)+1 = m(\xi(n,w-1)+1),$$

as required.

Now, to prove the claim, fix a surviving state space $i \in G$ of $B|_{\overline{G}\leftarrow U}$ other than the last surviving state space (which can always be assumed to have width 2 anyway). State space i is followed by $m-1$ restricted layers. As in Lemma 3.4, with probability at least $1-2^{1-m}$, at least one of these layers will reduce the width (by selecting a pair of edges causing a collision). This reduces the number of reachable vertices and, hence, the width of state space i. A union bound gives the required probability. □

Lemma 6.4. *For all $n \geq 1$, we have $\xi(n,2) \leq 5$.*

Proof. Let B be a length-n, width-2, ordered branching program. We may assume, without loss of generality, that B has no trivial layers, as trivial layers can be removed without affecting the other Fourier coefficients. As in Lemma 3.21, we can do a case analysis over all possible nontrivial layers, to see that $\lambda(B_i) \leq 1/2$ for all $i \in [n]$. We also have $\|\widehat{B_i}[0]\|_2 \leq \sqrt{2}$ (since $\widehat{B_i}[0]$ is a 2×2 stochastic matrix) and $\|\widehat{B_i}[1]\|_2 \leq 1$ for all $i \in [n]$. As in Proposition 3.6, we can now inductively bound the damped Fourier mass using Lemmas 2.1 and 3.9.

$$\begin{aligned}L_p(B_{1\cdots i+1}) \leq & L_p(B_{1\cdots i})L_p(B_{i+1}) + L_p(B_{1\cdots i})\lambda(B_{i+1}) + \left\|\widehat{B_{1\cdots i}}[0]\right\|_2 L_p(B_{i+1}) \\ \leq & L_p(B_{1\cdots i})p + L_p(B_{1\cdots i})\frac{1}{2} + \sqrt{2}p\,.\end{aligned}$$

Setting $p = 1/5$ shows that, if $L_p(B_{1\cdots i}) \leq 1$, then $L_p(B_{1\cdots i+1}) \leq 1$. Induction then gives $L_{1/5}(B) \leq 1$. This implies that $L^1(B) \leq 5$, as required. □

Solving the recurrence for ξ gives $\xi(n,w) \leq O(\log n)^{w-2}$, as required for Theorem 6.1.

7 Further directions

7.1 Larger width

Our results hinge on the fact that "mixing" is well understood for regular ordered branching programs [9, 36, 28, 14, 45] and for (non-regular) width-2 ordered branching programs [5]. Indeed, understanding mixing underpins most results for restricted models of branching programs.

The other main ingredient in our proof is using random restrictions to reduce from width 3 to "almost" width 2 (Section 3.1). After applying the random restriction, we can exploit our understanding of mixing for width-2 (Section 3.2).

What about width 4 and beyond? Using a random restriction we can reduce analyzing width 4 to "almost" width 3—that is, Proposition 3.2 generalizes. Unfortunately, the reduction does not give a true width-3, ordered branching program, but rather a concatenation of width-4 programs that begin and end in width-3 state spaces and have few nonregular layers; thus we cannot repeat the reduction to width 2. Moreover, we have a poor understanding of mixing for non-regular width-3, ordered branching programs, which means we cannot use the same techniques that have worked for width-2, ordered branching programs. Indeed, the biggest obstacle to extending our techniques to $w > 3$ is mixing (Lemma 3.10). The problem is that the parameter $\lambda(D)$ is no longer a useful measure of mixing for width-3 and above. In particular, $\lambda(D) > 1$ is possible if $\mathbb{E}_U[D[U]]$ is a 3×3 matrix. To extend our techniques, we need a better notion of mixing. Using $\lambda(D)$ is useful for regular ordered branching programs (it equals the second eigenvalue for regular programs), but is of limited use for non-regular ordered branching programs. One idea is to use the 1-norm, rather than the 2-norm. Define

$$\lambda_1(D) := \max_{\sum_u v(u)=0} \frac{\|v\mathbb{E}_U[D[U]]\|_1}{\|v\|_1}. \tag{7.1}$$

This would still satisfy $\lambda_1(D) \in [0,1]$ even for non-regular D of arbitrary width. The partition lemma (Lemma 3.16) would also generalize to larger width with this definition. However, we do not know an analog of Lemmas 3.21 or 3.22 for larger width.

Our proof also uses a different notion of mixing—collisions. Namely, to prove Proposition 3.2, we used the fact that a random restriction of a non-regular layer will with probability at least 1/2 result in the width of the right side of the layer being reduced. This is a form of mixing, but it is not captured by λ. Ideally, we want a notion of mixing that captures both λ and width-reduction under restrictions.

Our proofs combine the techniques of Braverman et al. [9] and Reingold et al. [36] and those of Brody and Verbin [10] and Steinberger [44]. We would like to combine them more cleanly—presently the proof is split into two parts (Proposition 3.2 and Lemma 3.10). This would likely involve developing a deeper understanding of the notion of mixing.

7.2 Better seed length

Our seed length $\widetilde{O}(\log^3 n)$ is far from the optimal $O(\log n)$. Further improvement would require some new techniques. We could potentially relax our notion of Fourier growth to achieve better results. We bound $L^k(f)$ in order to obtain a bound on $L(f|_t)$ for an averaging restriction t, whence we can apply a small-bias generator to $f|_t$. However, it suffices to bound $L(g)$, where g *approximates* $f|_t$, as stated below.

Proposition 7.1 (De et al. [15, Proposition 2.6]). *Let $f, f_+, f_- : \{0,1\}^n \to \mathbb{R}$ satisfy $f_-(x) \le f(x) \le f_+(x)$ for all x and $\mathbb{E}_U[f_+(U) - f_-(U)] \le \delta$. Then any ε-biased distribution X gives*

$$\left|\mathop{\mathbb{E}}_X[f(X)] - \mathop{\mathbb{E}}_U[f(U)]\right| \le \delta + \varepsilon \cdot \max\{L(f_+), L(f_-)\}.$$

The functions f_+ and f_- are called sandwiching polynomials for f. This notion of sandwiching is in fact a tight characterisation of small bias [15, Proposition 2.7]. That is, any function f fooled by all small bias generators has sandwiching polynomials satisfying the hypotheses of Proposition 7.1.

Gopalan et al. [18] use sandwiching polynomials in the analysis of their generator for CNFs. This allows them to set a constant fraction of the bits at each level of recursion ($p = \Omega(1)$), while we set a $p = 1/O(\log n)$ fraction at each level. We would like to similarly exploit sandwiching polynomials for branching programs to improve the seed length of the generator.

A further avenue for improvement (also exploited by Gopalan et al.) would be to modify the generator construction to have polyloglog$(n)/p$ levels of recursion, rather than $\Theta(\log(n)/p)$. Gopalan et al. analyze this by showing that read-once CNFs simplify substantially after restricting all but a $1/\text{polylog}(n)$ fraction of the variables. Unfortunately, we do not know of an appropriate notion of simplification to use for arbitrary width-3 programs.

References

[1] ANIL ADA, OMAR FAWZI, AND HAMED HATAMI: Spectral norm of symmetric functions. In *Proc. 16th Internat. Workshop on Randomization and Computation (RANDOM'12)*, volume 7408 of *LNCS*, pp. 338–349. Springer, 2012. [doi:10.1007/978-3-642-32512-0_29, arXiv:1205.5282] 4

[2] NOGA ALON, ODED GOLDREICH, JOHAN HÅSTAD, AND RENÉ PERALTA: Simple constructions of almost k-wise independent random variables. *Random Structures Algorithms*, 3(3):289–304, 1992. Preliminary version in FOCS'90. [doi:10.1002/rsa.3240030308] 4, 12

[3] MIHIR BELLARE AND JOHN ROMPEL: Randomness-efficient oblivious sampling. In *Proc. 35th FOCS*, pp. 276–287. IEEE Comp. Soc. Press, 1994. [doi:10.1109/SFCS.1994.365687] 36

[4] MICHAEL BEN-OR AND NATHAN LINIAL: Collective coin flipping. *Advances in Comput. Research*, 5:91–115, 1989. Available at author's website. Preliminary version in FOCS'85. 5, 6, 38

[5] ANDREJ BOGDANOV, ZEEV DVIR, ELAD VERBIN, AND AMIR YEHUDAYOFF: Pseudorandomness for width-2 branching programs. *Theory of Computing*, 9(7):283–293, 2013. [doi:10.4086/toc.2013.v009a007] 3, 4, 12, 43

[6] ANDREJ BOGDANOV, PERIKLIS A. PAPAKONSTANTINOU, AND ANDREW WAN: Pseudorandomness for read-once formulas. In *Proc. 52nd FOCS*, pp. 240–246. IEEE Comp. Soc. Press, 2011. [doi:10.1109/FOCS.2011.57] 3, 11

[7] JEAN BOURGAIN: On the distribution of the Fourier spectrum of Boolean functions. *Israel J. Mathematics*, 131(1):269–276, 2002. [doi:10.1007/BF02785861] 5

[8] YIGAL BRANDMAN, ALON ORLITSKY, AND JOHN L. HENNESSY: A spectral lower bound technique for the size of decision trees and two level AND/OR circuits. *IEEE Trans. Computers*, 39(2):282–287, 1990. [doi:10.1109/12.45216] 4

[9] MARK BRAVERMAN, ANUP RAO, RAN RAZ, AND AMIR YEHUDAYOFF: Pseudorandom generators for regular branching programs. *SIAM J. Comput.*, 43(3):973–986, 2014. Preliminary version in FOCS'10. [doi:10.1137/120875673] 3, 21, 25, 43, 44

[10] JOSHUA BRODY AND ELAD VERBIN: The coin problem and pseudorandomness for branching programs. In *Proc. 51st FOCS*, pp. 30–39. IEEE Comp. Soc. Press, 2010. [doi:10.1109/FOCS.2010.10] 3, 5, 6, 7, 13, 15, 41, 44

[11] JEHOSHUA BRUCK: Harmonic analysis of polynomial threshold functions. *SIAM J. Discrete Math.*, 3(2):168–177, 1990. [doi:10.1137/0403015] 4

[12] JEHOSHUA BRUCK AND ROMAN SMOLENSKY: Polynomial threshold functions, AC^0 functions, and spectral norms. *SIAM J. Comput.*, 21(1):33–42, 1992. Preliminary version in FOCS'90. [doi:10.1137/0221003] 4

[13] L. ELISA CELIS, OMER REINGOLD, GIL SEGEV, AND UDI WIEDER: Balls and bins: Smaller hash families and faster evaluation. *SIAM J. Comput.*, 42(3):1030–1050, 2013. Preliminary version in FOCS'11. [doi:10.1137/120871626] 2

[14] ANINDYA DE: Pseudorandomness for permutation and regular branching programs. In *Proc. 26th IEEE Conf. on Computational Complexity (CCC'11)*, pp. 221–231. IEEE Comp. Soc. Press, 2011. [doi:10.1109/CCC.2011.23] 3, 43

[15] ANINDYA DE, OMID ETESAMI, LUCA TREVISAN, AND MADHUR TULSIANI: Improved pseudorandom generators for depth 2 circuits. In *Proc. 14th Internat. Workshop on Randomization and Computation (RANDOM'10)*, volume 6302 of *LNCS*, pp. 504–517. Springer, 2010. [doi:10.1007/978-3-642-15369-3_38] 44

[16] LARS ENGEBRETSEN, PIOTR INDYK, AND RYAN O'DONNELL: Derandomized dimensionality reduction with applications. In *Proc. 13th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'02)*, pp. 705–712. ACM/SIAM, 2002. ACM DL. 2

[17] EHUD FRIEDGUT: Boolean functions with low average sensitivity depend on few coordinates. *Combinatorica*, 18(1):27–35, 1998. [doi:10.1007/PL00009809] 5

[18] PARIKSHIT GOPALAN, RAGHU MEKA, OMER REINGOLD, LUCA TREVISAN, AND SALIL P. VADHAN: Better pseudorandom generators from milder pseudorandom restrictions. In *Proc. 53rd FOCS*, pp. 120–129. IEEE Comp. Soc. Press, 2012. [doi:10.1109/FOCS.2012.77, arXiv:1210.0049] 3, 4, 5, 31, 32, 44

[19] BEN GREEN AND TOM SANDERS: Boolean functions with small spectral norm. *Geometric and Functional Analysis*, 18(1):144–162, 2008. [doi:10.1007/s00039-008-0654-y] 4

[20] VINCE GROLMUSZ: On the power of circuits with gates of low ℓ_1 norms. *Theoret. Comput. Sci.*, 188(1-2):117–128, 1997. [doi:10.1016/S0304-3975(96)00290-3] 4

[21] IFTACH HAITNER, DANNY HARNIK, AND OMER REINGOLD: On the power of the randomized iterate. *SIAM J. Comput.*, 40(6):1486–1528, 2011. Preliminary version in CRYPTO'06. [doi:10.1137/080721820] 2

[22] JOHAN HÅSTAD: Almost optimal lower bounds for small depth circuits. In *Proc. 18th STOC*, pp. 6–20. ACM Press, 1986. [doi:10.1145/12130.12132] 7

[23] ALEXANDER HEALY, SALIL VADHAN, AND EMANUELE VIOLA: Using nondeterminism to amplify hardness. *SIAM J. Comput.*, 35(4):903–931, 2006. Preliminary version in STOC'04. [doi:10.1137/S0097539705447281] 2

[24] RUSSELL IMPAGLIAZZO, RAGHU MEKA, AND DAVID ZUCKERMAN: Pseudorandomness from shrinkage. In *Proc. 53rd FOCS*, pp. 111–119. IEEE Comp. Soc. Press, 2012. [doi:10.1109/FOCS.2012.78] 3, 4

[25] RUSSELL IMPAGLIAZZO, NOAM NISAN, AND AVI WIGDERSON: Pseudorandomness for network algorithms. In *Proc. 26th STOC*, pp. 356–364. ACM Press, 1994. [doi:10.1145/195058.195190] 3

[26] PIOTR INDYK: Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, 2006. Preliminary version in FOCS'00. [doi:10.1145/1147954.1147955] 2

[27] EYAL KAPLAN, MONI NAOR, AND OMER REINGOLD: Derandomized constructions of k-wise (almost) independent permutations. *Algorithmica*, 55(1):113–133, 2009. Preliminary versions in RANDOM'05 and ECCC. [doi:10.1007/s00453-008-9267-y] 2

[28] MICHAL KOUCKÝ, PRAJAKTA NIMBHORKAR, AND PAVEL PUDLÁK: Pseudorandom generators for group products. In *Proc. 43rd STOC*, pp. 263–272. ACM Press, 2011. Preliminary version in ECCC. [doi:10.1145/1993636.1993672] 3, 43

[29] EYAL KUSHILEVITZ AND YISHAY MANSOUR: Learning decision trees using the Fourier spectrum. *SIAM J. Comput.*, 22(6):1331–1348, 1993. Preliminary version in STOC'91. [doi:10.1137/0222080] 4

[30] YISHAY MANSOUR: An $O(n^{\log\log n})$ learning algorithm for DNF under the uniform distribution. *J. Comput. System Sci.*, 50(3):543–550, 1995. Preliminary version in COLT'92. [doi:10.1006/jcss.1995.1043] 4, 5

[31] JOSEPH NAOR AND MONI NAOR: Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993. Preliminary version in STOC'90. [doi:10.1137/0222053] 4, 12

[32] JELANI NELSON: Exponential tail bounds without the moment generating function. MathOverflow, 2013. http://mathoverflow.net/q/147239. 36

[33] NOAM NISAN: $\mathcal{RL} \subseteq \mathcal{SC}$. *Comput. Complexity*, 4(1):1–11, 1994. Preliminary version in STOC'92. [doi:10.1007/BF01205052] 2

[34] NOAM NISAN AND DAVID ZUCKERMAN: More deterministic simulation in logspace. In *Proc. 25th STOC*, pp. 235–244. ACM Press, 1993. [doi:10.1145/167088.167162] 3

[35] OMER REINGOLD: Undirected connectivity in log-space. *J. ACM*, 55(4):17:1–17:24, 2008. Preliminary version in STOC'05. [doi:10.1145/1391289.1391291] 7, 18

[36] OMER REINGOLD, THOMAS STEINKE, AND SALIL P. VADHAN: Pseudorandomness for regular branching programs via Fourier analysis. In *Proc. 17th Internat. Workshop on Randomization and Computation (RANDOM'13)*, volume 8096 of *LNCS*, pp. 655–670. Springer, 2013. [doi:10.1007/978-3-642-40328-6_45, arXiv:1306.3004] 3, 4, 5, 6, 7, 20, 21, 22, 31, 32, 33, 43, 44

[37] OMER REINGOLD, LUCA TREVISAN, AND SALIL P. VADHAN: Pseudorandom walks on regular digraphs and the $\mathcal{RL}$ vs. $\mathcal{L}$ problem. In *Proc. 38th STOC*, pp. 457–466. ACM Press, 2006. [doi:10.1145/1132516.1132583] 7, 18

[38] EYAL ROZENMAN AND SALIL P. VADHAN: Derandomized squaring of graphs. In *Proc. 9th Internat. Workshop on Randomization and Computation (RANDOM'05)*, volume 3624 of *LNCS*, pp. 436–447. Springer, 2005. [doi:10.1007/11538462_37] 18

[39] MICHAEL E. SAKS AND SHIYU ZHOU: $\mathrm{BP}_H\mathrm{SPACE}(\mathrm{S}) \subseteq \mathrm{DSPACE}(\mathrm{S}^{3/2})$. *J. Comput. System Sci.*, 58(2):376–403, 1999. Preliminary version in FOCS'95. [doi:10.1006/jcss.1998.1616] 3, 4

[40] JEANETTE P. SCHMIDT, ALAN SIEGEL, AND ARAVIND SRINIVASAN: Chernoff-Hoeffding bounds for applications with limited independence. *SIAM J. Discrete Math.*, 8(2):223–250, 1995. Preliminary version in SODA'93. [doi:10.1137/S089548019223872X] 35, 36

[41] AMIR SHPILKA, AVISHAY TAL, AND BEN LEE VOLK: On the structure of boolean functions with small spectral norm. *Comput. Complexity*, 26(1):229–273, 2017. Preliminary version in ITCS'14. [doi:10.1007/s00037-015-0110-y, arXiv:1304.0371] 4

[42] JIŘÍ ŠÍMA AND STANISLAV ŽÁK: A sufficient condition for sets hitting the class of read-once branching programs of width 3. In *Internat. Conf. Theory and Practice of Comput. Sci. (SOFSEM'12)*, pp. 406–418, 2012. [doi:10.1007/978-3-642-27660-6_33] 3

[43] D. SIVAKUMAR: Algorithmic derandomization via complexity theory. In *Proc. 34th STOC*, pp. 619–626. ACM Press, 2002. Also in CCC'02. [doi:10.1145/509907.509996] 2

[44] JOHN P. STEINBERGER: The distinguishability of product distributions by read-once branching programs. In *Proc. 28th IEEE Conf. on Computational Complexity (CCC'13)*, pp. 248–254. IEEE Comp. Soc. Press, 2013. [doi:10.1109/CCC.2013.33] 7, 13, 14, 15, 41, 42, 44

[45] THOMAS STEINKE: Pseudorandomness for permutation branching programs without the group theory. *Electron. Colloq. on Comput. Complexity (ECCC)*, 19(83), 2012. Available at ECCC. 3, 7, 43

[46] THOMAS STEINKE, SALIL VADHAN, AND ANDREW WAN: Pseudorandomness and Fourier growth bounds for width-3 branching programs. In *Proc. 18th Internat. Workshop on Randomization and Computation (RANDOM'14)*, volume 28 of *LIPIcs*, pp. 885–899. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2014. Preliminary version in ECCC. [doi:10.4230/LIPIcs.APPROX-RANDOM.2014.885, arXiv:1405.7028] 1

[47] HING YIN TSANG, CHUNG HOI WONG, NING XIE, AND SHENGYU ZHANG: Fourier sparsity, spectral norm, and the log-rank conjecture. In *Proc. 54th FOCS*, pp. 658–667. IEEE Comp. Soc. Press, 2013. [doi:10.1109/FOCS.2013.76, arXiv:1304.1245] 4

[48] YOAV TZUR: Notions of weak pseudorandomness and GF(2^n)-polynomials. Master's thesis, Weizmann Institute, 2009. Available at ECCC. 3

AUTHORS

Thomas Steinke
Postdoctoral researcher
IBM Almaden Research Center
San Jose, CA
width3@thomas-steinke.net
http://www.thomas-steinke.net

Salil Vadhan
Professor
John A. Paulson School of Engineering and Applied Sciences
Harvard University, Cambridge, MA
salil@seas.harvard.edu
http://www.seas.harvard.edu/~salil

Andrew Wan
Research staff member
Institute for Defense analyzes, Alexandria, VA
atw12@columbia.edu
http://www1.cs.columbia.edu/~atw12

ABOUT THE AUTHORS

THOMAS STEINKE is a postdoctoral researcher at the IBM Almaden Research Center. He completed his Ph. D. at Harvard University in 2016 advised by Salil Vadhan and was previously an undergraduate student at the University of Canterbury in New Zealand. His research interests include pseudorandomness, data privacy, and adaptive data analysis. When not proving theorems, he enjoys rock climbing with his wife Joy.

SALIL VADHAN is the Vicky Joseph Professor of Computer Science and Applied Mathematics at the Harvard John A. Paulson School of Engineering & Applied Sciences. He received his Ph. D. under the supervision of Shafi Goldwasser at MIT in 1999; his dissertation was entitled "A Study of Statistical Zero-Knowledge Proofs." Other research interests include the theory of pseudorandomness and the theory and practice of data privacy. He enjoys spending leisure time with his wife and two daughters, as well as learning to surf when the Boston weather or travel destinations permit.

ANDREW WAN is currently a research staff member at the Institute for Defense analyzes. Before this, he was a research fellow at the Simons Institute for the Theory of Computing, a postdoctoral fellow at Harvard University, and an assistant professor at IIIS, Tsinghua University. He received his doctorate from Columbia University in 2010 under the supervision of Tal Malkin and Rocco Servedio. His interests include complexity theory, cryptography, and machine learning. Before graduate school, he was a student of philosophy at Columbia University and enjoyed playing the piano, the trumpet, and the accordion. Although he still enjoys playing music, the PAC model rarely affords him the time.