Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 171 (2017)

Improved Pseudorandomness for Unordered Branching Programs through Local Monotonicity

Eshan Chattopadhyay[*]
Cornell University and IAS

Pooya Hatami[†]
University of Texas at Austin

Omer Reingold[‡]
Stanford University

Avishay Tal[§]
Stanford University

November 8, 2017

Abstract

We present an explicit pseudorandom generator with seed length $\widetilde{O}((\log n)^{w+1})$ for read-once, oblivious, width w branching programs that can read their input bits in any order. This improves upon the work of Impaggliazzo, Meka and Zuckerman (FOCS'12) where they required seed length $n^{1/2+o(1)}$.

A central ingredient in our work is the following bound that we prove on the Fourier spectrum of branching programs. For any width w read-once, oblivious branching program $B : \{0,1\}^n \to \{0,1\}$, any $k \in \{1,\ldots,n\}$,

$$\sum_{S\subseteq[n]:|S|=k} |\widehat{B}(S)| \leq O(\log n)^{wk}.$$

This settles a conjecture posed by Reingold, Steinke, and Vadhan (RANDOM'13).

Our analysis crucially uses a notion of local monotonicity on the edge labeling of the branching program. We carry critical parts of our proof under the assumption of local monotonicity and show how to deduce our results for unrestricted branching programs.

[*] eshanc@ias.edu. Supported by NSF grant CCF-1412958 and the Simons Foundation.

[†] pooyahat@gmail.com. Supported by a Simons Investigator Award (#409864, David Zuckerman)

[‡] reingold@stanford.edu. Supported in part by NSF grant CCF-1749750.

[§] avishay.tal@gmail.com. Supported by a Motwani Postdoctoral Fellowship and by NSF grant CCF-1749750.

ISSN 1433-8092

1 Introduction

1.1 Pseudorandom Generators for Branching Programs

A central goal in the area of pseudorandomness is to show that randomized algorithms are not much more powerful than deterministic algorithms. More precisely, the two main challenges are to show that randomness cannot save time and cannot save memory. It is known that showing that every polynomial time randomized algorithm can be made deterministic with only polynomial slowdown, and thus proving **BPP** = **P**, requires proving circuit lower bounds that seem much beyond the reach of current techniques. However, there are no known fundamental barriers known for proving that every randomized algorithm using space s can be made deterministic using space $O(s)$. This, by standard padding arguments, is equivalent to proving that $\mathbf{RL} = \mathbf{L}$.

The main tool in derandomizing **RL** is a pseudorandom generator (PRG) that takes a seed of length d and stretches it to n bits such that no logarithmic-space algorithm can distinguish it from a uniform distribution on n bits with significant advantage. More concretely, to prove that $\mathbf{RL} = \mathbf{L}$, it is enough to construct a pseudorandom generator $G : \{0,1\}^d \to \{0,1\}^n$ that is computable in space $O(\log n)$ and satisfies: (a) the seed length d is $O(\log n)$, and (b) $G(U_d)$ is indistinguishable from U_n for randomized algorithms that use $O(\log n)$ space. Apart from its direct derandomization implication, such a generator G would also have a variety of other applications [Ind06, CRSW13, EIO02, Siv02, HVV06, HHR11].

An ordered branching program[1] B of width w is a non-uniform model of computation, and captures randomized algorithms with space $\lfloor \log w \rfloor$. Thus, it is enough to construct optimal PRGs for ordered branching programs with width poly(n) to derandomize **RL**. A branching program B maintains a state in the set $\{1, \ldots, w\}$ and reads the input bits in a known fixed order. At time step $i = 1, \ldots, n$, B reads a bit and uses a transition function $\Gamma_i : [w] \times \{0,1\} \to [w]$ to move to a new state. Thus, B can be thought of as a layered directed graph, with w nodes in each layer, and two edges going out of each node to the immediately next layer, one labelled with a 0 and the other labelled with a 1.

A well known PRG construction of Nisan [Nis92] achieves seed length $d = O(\log^2 n)$ for polynomial width ordered branching programs, and was used by Saks and Zhou [SZ99] in their proof that $\mathbf{RL} \subseteq \mathbf{L}^{3/2}$. However, despite a lot of effort it is still open to improve the seed length of Nisan's generator even when we restrict the distinguishers to have less than 2 bits of space (i.e., ordered width 3 branching programs). There have been improvements to Nisan's generator for various special classes of branching programs [SZ95, BRRY10, BV10, KNP11, De11, RSV13, SVW14]. Most of these works are based on a more refined analysis of [Nis92] or of closely related constructions [INW94, NZ96]. The basic idea of this framework is that if T_1 and T_2 are two consecutive time intervals, and width w branching program B uses ℓ random bits in the interval T_1, then it remembers at most $\log w$ bits of the information about the randomness used in T_1. Thus, we can use a seeded extractor to extract out and re-use $(\ell - \log w)$ bits in T_2. A major bottleneck with all known analysis done in this framework is that it is heavily dependent on the ordering of the bits (T_1 and T_2 cannot be interleaved).

[1] in this work, by a "branching program", we refer to an "oblivious read-once branching program"

A natural question to ask is the construction of a PRG that is not sensitive to the ordering of the bits in which the input is read, i.e., fooling the class of (read-once oblivious) branching programs that read its input in an unknown (but fixed) order. Such a PRG would need completely new ideas since it is not clear if the above mentioned framework works in this setting (in fact, Tzur [Tzu09] proved that Nisan's PRG can in fact be distinguished from uniform by an unordered constant width branching program). The hope is that PRGs that are not sensitive to the ordering would help make progress on the original problem of fooling ordered branching programs using seed length $o(\log^2 n)$.

Developing PRGs for unordered branching programs have attracted attention recently. Bogdanov, Papakonstantinou, and Wan [BPW11] gave an explicit construction of a PRG for width $w = 2^{\Omega(n)}$ with seed length $(1 - \Omega(1))n$. Impagliazzo, Meka, and Zuckerman [IMZ12] gave a PRG with the seed length to $(nw)^{1/2+o(1)}$ (they actually achieve stronger results, and show how to fool arbitrary branching programs of size s that may read their input bits more than once, using seed length $s^{1/2+o(1)}$). Reingold, Steinke and Vadhan [RSV13] achieve a seed-length of $O(w^2 \log^2 n)$ for the restricted class of permutation branching programs. Finally, Steinke, Vadhan and Wan [SVW14] constructed a PRG for width 3 unordered branching programs with seed-length $\widetilde{O}(\log^3 n)$.

Our approach to constructing PRGs for unordered branching programs is based on a construction proposed by Ajtai and Wigderson [AW85] for fooling constant depth circuits, and was used more recently by Gopalan et al. [GMR$^+$12]. Informally, the construction is the following: Pseudorandomly partition the coordinates $\{1, \dots, n\}$, and use an independent copy of an ε-biased generator [NN93] for each part. Based on this, [GMR$^+$12] constructed PRGs for combinatorial rectangles, read-once CNFs and hitting set generators for width 3 branching programs with seed length $\widetilde{O}(\log n)$ and polynomially small error.

A crucial insight in the work of Gopalan et al. is the following: Let S be a block in the partition and let $\overline{S} = [n] \setminus S$. Further, let $f : \{0,1\}^n \to \{0,1\}$ be the function we are trying to fool. Let $f(x, y)$ denote f evaluated on an n bit string with coordinates in S set to x and coordinates in $\overline{S}$ set to y (where $x \in \{0,1\}^S$, and $y \in \{0,1\}^{\overline{S}}$). Then, it is enough to show that the function $g(x) = \mathbf{E}_{y \sim U_{\overline{S}}}[f(x, y)]$ can be fooled by an ε-biased generator[2]. The hope is that the function f is much easier to fool on 'average'.

The works mentioned above of Reingold, Steinke, and Vadhan [RSV13] and Steinke, Vadhan and Wan [SVW14] in fact show that the generator of Gopalan et al. fools the respective class of unordered branching programs, and the main tool they use to prove this is Fourier analysis of branching programs.

We obtain our result on PRGs for unordered branching programs by further extending the approaches of [RSV13, SVW14], and a central component of our work is an affirmative answer to a conjecture on the Fourier spectrum of branching programs posed in [RSV13].

The following is our main result on PRGs for unordered branching programs.

Theorem 1. *For all $n > 0$ and $w \leq \log n$, there exists an explicit pseudorandom generator for the class of unordered, read-once, oblivious branching programs of length n and width w with seed length $O((\log^{w+1} n) \log \log n)$ and error $1/n^{O(1)}$.*

[2] this is not entirely accurate, and one requires more structure on f. We refer the reader to Section 6 for more details.

1.2 On the Fourier Spectrum of Branching Programs

For any Boolean function $f : \{0,1\}^n \to \{0,1\}$, and any set $S \subseteq [n]$, define the Fourier coefficient $\widehat{f}(S) := \mathbf{E}_{x \in \{0,1\}^n}[f(x) \cdot (-1)^{\sum_{i \in S} x_i}]$ (see Section 3.2 for more details on Fourier analysis of Boolean functions). A central complexity measure of f is given by the spectral norm: $L_1(f) = \sum_{S \neq \emptyset} |\widehat{f}(S)|$. It is well known that a δ-biased generator with $\delta = \varepsilon / L_1(f)$ will ε-fool f. Further, bounds on $L_1(f)$ for specific classes of functions have been extensively studied in the literature with applications to learning theory, communication complexity and circuit complexity. It is known that the spectral norm of width 2 branching programs is bounded by $O(n)$ [SZ95, BDVY13], and thus can be fooled by ε-biased generators. However, the Mod 3 function (that equals 1 iff the Hamming weight of the input is divisible by 3) which is computable by a width 3 branching program has exponential spectral norm.

It turns out that in the context of analyzing the generator of Gopalan et al. [GMR$^+$12], the more relevant quantity to look at is $L_{1,k}(f) = \sum_{S:|S|=k} |\widehat{f}(S)|$, for $k \in [n]$. Reingold et al. [RSV13] proved the following result: Let $\mathcal{C}$ be any class of branching program that are closed under restrictions and decompositions. Further suppose that for $f \in \mathcal{C}$, and we have $L_{1,k}(f) \leq \text{poly}(n) \cdot t^k$ for all $k \in [n]$. Then the generator of Gopalan et al. can be used to achieve seed length $\widetilde{O}(t \log^2 n)$. Moreover, [RSV13] proved that when f is a regular branching program of width w, then $L_{1,k}(f) \leq (2w^2)^k$. Based on these estimates, they obtained PRGs for permutation branching programs with seed length $\widetilde{O}(w^2 \log^2 n)$.

Further, [RSV13] conjectured that for any width w branching program $L_{1,k}(f) \leq \text{poly}(n)(\log n)^{O_w(k)}$ for all $k \in [n]$. This was confirmed for $w = 3$ by Steinke et al. [SVW14], and they obtained a PRG for width 3 unordered branching programs with seed length $\widetilde{O}(\log^3 n)$. The case $k = 1$ was settled for all widths w by Steinke et al. [SVW14] extending the techniques introduced in [Ste13].

Our main contribution here is the following theorem that settles the conjecture posed by Steinke, Reingold and Vadhan ([RSV13, Conjecture 8.1]).

Theorem 2. *Let B be an ordered read-once, oblivious branching program of length n and width w. Then,*

$$\sum_{s:|s|=k} \left| \widehat{B}(s) \right| \leq O(\log n)^{wk} ,$$

for all $k \in [n]$.

A direct application of Theorem 2 and invoking the work of [RSV13] would imply a PRG with seed length $\widetilde{O}(\log^{w+2} n)$ for unordered branching programs with width w. We improve the seed length to $\widetilde{O}(\log^{w+1} n)$ by sharpening the estimates in Theorem 2 assuming that every node in the branching program is reachable by a random walk (from the start node) on the branching program with probability more than $1/n^{O(1)}$. We show that indeed in order to fool general branching program, it is sufficient to fool this restricted class of branching programs. While this class of branching programs is not closed under restrictions and sub-programs, we prove that the generator of Gopalan et al. [GMR$^+$12] still works.

2 Proof Techniques

In this section, we present that main ideas in the proof of Theorem 2, bounding $L_{1,k}(B) = \sum_{s:|s|=k} |\widehat{B}(s)|$ for all branching programs of width-w and length-n. We first mention the case $k = 1$, that was proved by Steinke et al. [SVW14] following Steinberger's approach [Ste13]. In this case $\sum_{s:|s|=1} |\widehat{B}(s)|$ is just the total effect of the function, as defined next.

Effects and Influences. The effect of bit i on a Boolean function $B : \{0,1\}^n \to \{0,1\}$ is

$$\mathrm{Eff}_i(B) := |\mathbf{Pr}[B(x) = 1 \mid x_i = 1] - \mathbf{Pr}[B(x) = 1 \mid x_i = 0]| \,.$$

It is worth-while to compare the effect of a variable to the (well-known) influence of a variable:

$$\mathrm{Inf}_i(B) := \mathop{\mathbf{Pr}}_{x \in \{0,1\}^n}[B(x) \neq B(x \oplus e_i)],$$

where e_i is the i-th standard unit vector. It is easy to observe that $\mathrm{Eff}_i(B) \le \mathrm{Inf}_i(B)$ for any Boolean function B, as explained by the following interpretation of effects and influences when we view B as a voting scheme. The effect of voter i is its probability to affect the decision (the value of $B(x)$) if he casts his vote first, and all the other voters vote after him uniformly at random; the influence is its probability to affect the decision if he casts his vote last and can vote adaptively after seeing all the other votes. For example, if B is the parity function, then every voter has influence 1 and effect 0. The total effect (total influence, resp.) of B is defined as $\sum_{i=1}^{n} \mathrm{Eff}_i(B)$ ($\sum_{i=1}^{n} \mathrm{Inf}_i(B)$, resp.).

Both effects and influences have nice expressions in terms of the Fourier transform of B: $\mathrm{Eff}_i(B) = 2 \cdot |\widehat{B}(\{i\})|$ and $\mathrm{Inf}_i(B) = \sum_{s \ni i} \widehat{B}(s)^2$.

The Coin-Problem. Before explaining our results, we take a detour to the works of Brody and Verbin [BV10] and Steinberger [Ste13], who studied the coin-problem for bounded-width branching programs. In the coin problem, the branching program tries to distinguish a sequence of n independent unbiased coins from a sequence of n independent coins with probabilities $(1/2-\varepsilon, 1/2+\varepsilon)$. This line of work showed that oblivious read-once branching programs (ROBPs in short) of width-w and length-n can distinguish a uniformly random string from a $(1/2-\varepsilon, 1/2+\varepsilon)$-biased random string only if $\varepsilon \ge 1/O(\log n)^{w-2}$. Furthermore, this result is tight up to the constant hidden in the big-Oh notation.

Brody and Verbin [BV10] observed that without loss of generality the branching-program can be assumed to be "locally monotone" (or "weakly monotone"), to be defined shortly. Moreover, Brody and Verbin [BV10] showed that locally monotone branching-programs have a very nice structural dichotomy that allows one to reduce width with high probability after applying a random restriction. To introduce local monotonicity and the structural dichotomy, let us set some general notations for branching programs first.

Branching Programs. A branching program B of length n and width w is a directed layered graph with $n+1$ layers of vertices denoted $V_1, \ldots, V_{n+1}$. Each V_i (except V_1) consists of w vertices $\{v_{i,1}, \ldots, v_{i,w}\}$, and between every two consecutive layers V_i and V_{i+1} there exists a set of directed edges (from V_i to V_{i+1}), denoted E_i, such that any vertex in V_i has precisely two out-going edges in E_i, one marked by 1 and one marked by 0. V_1 consists of a single vertex, denoted $v_{1,1}$. V_{n+1} vertices are marked with either 'accept' and 'reject'.

A branching program B and an input $x \in \{0,1\}^n$ naturally describes a computation path in the layered graph: we start at node $v_1 = v_{1,1}$ in V_1. For $i = 1, \dots, n$, we traverse according to the edge touching v_i in E_i marked by x_i to get to a node $v_{i+1} \in V_{i+1}$. The resulting computational path is $v_1 \to v_2 \to \dots \to v_{n+1}$. We say that B accepts x iff the computational path defined by B and x reaches an accepting node. Naturally B describes a Boolean function $B : \{0,1\}^n \to \{0,1\}$ whose value is 1 on input x iff B accepts x.

In Section 6, we may assume that the bits of x are permuted by a permutation $\pi \in S_n$. In such a case, the bits of x are read according to π, i.e., in the i-th layer of the programs we follow the edge marked by x_{π_i}. We denote the resulting Boolean function by $B^{\pi}(x) := B(x_{\pi_1}, \dots, x_{\pi_n})$. For the Fourier bounds part, we may ignore the permutation, as $L_{1,k}(B) = L_{1,k}(B^{\pi})$ for any permutation π.

For a vertex $v \in V_i$ in the branching program we denote by $B_{\to v}$ the sub-branching program ending in the i-th layer and having v the only accepting state. We denote by $B_{v\to}$ the sub-branching program starting at v and ending at V_{n+1}. Observe that we may express the function computed by the branching program B as a sum of products of these sub-programs, namely

$$\forall i \in [n] : \forall x \in \{0,1\}^n : B(x) = \sum_{v \in V_i} B_{\to v}(x) \cdot B_{v\to}(x). \tag{1}$$

This decomposition will be very useful in our inductive argument. We denote by $p_v := \mathbf{Pr}[B_{\to v}(x) = 1]$ the probability that a random walk in B would reach v, and by $\beta_v := \mathbf{Pr}[B_{v\to}(x) = 1]$ the probability that a random walk starting from v would reach an accepting node.

2.1 Locally Monotone Branching Programs

Next, we define locally monotone branching programs. Let B be a width-w length-n ROBP. Since renaming the vertices in each layer does not affect the functionality of B, we may assume without loss of generality that the vertices in V_i are ordered according to β_v. That is, for every $i \in [n+1]$ and $j \in [w-1]$ we have $\beta_{v_{i,j}} \le \beta_{v_{i,j+1}}$. In the case that $\beta_{v_{i,j}} = \beta_{v_{i,j+1}}$ we break ties arbitrarily but commit to a strict ordering of the nodes in each layer. B is called locally monotone if for any vertex v in B the vertex reached from v using the 0-edge has lower or equal index than the vertex reached from v using the 1-edge.

Local monotonicity concerns only the labeling of the edges and not the structure of the graph. Thus, we believe that it is very different and arguably less restrictive than other previously studied notions such as permutation and regular branching programs [BRRY10, BV10, De11, RSV13].

There are many beautiful observations regarding this notion:

Programs can be locally monotonized. Any branching program can be locally monotonized by relabeling the edges. If some vertex v violates the local monotonicity we simply swap the labels of the two edges coming out of it. One may wonder in which order such a relabeling should be performed to ensure local monotonicity in the end (it is natural to go from the last layer back to the first), however surprisingly the order does not matter. This is due to the fact that any relabeling of edges does not change β_v since x is sampled from the uniform distribution and the probabilities only depend on the number of paths from v to the accepting nodes in V_{n+1} and not on the labeling of the edges. Thus, when deciding whether or not to relabel the edges touching $v \in V_i$, any relabeling that occurred in $E_1, \ldots, E_{i-1}, E_{i+1}, \ldots, E_n$ does not affect the decision.

Optimal for the coin-problem. As observed by [BV10], any distinguisher for the coin-problem can be assumed to be without loss of generality locally monotone. That is, if B is a distinguisher that distinguishes between n independent fair coins and n independent biased coins with advantage α, then the locally monotonized version of B distinguishes between the two distributions with advantage at least α.

Extremal for total effect. The total effect of any branching program B is at most the total effect of the locally monotonized version of B. Thus, bounding the total effect of locally monotone branching programs of width-w implies the same bound for all branching programs of width-w. (For total influence this is not the case, by considering the parity function.)

Colliding or identity layers. For $i \in [n]$, denote by E_i^0 the set of edges in E_i marked by 0 and similarly define E_i^1. We say that E_i is an identity layer if both E_i^0 and E_i^1 form the same matching between V_i and V_{i+1} (in which case x_i does not affect the output of of B). We say that E_i is a colliding layer if either E_i^0 or E_i^1 does not form a matching between V_i and V_{i+1} (i.e., there exists $b \in \{0,1\}$ and two edges in E_i^b that touches the same vertex in V_{i+1}). The following is a key-point in the works of [BV10, Ste13]:

Lemma 2.1 ([BV10])**.** *In a locally monotone branching program, each layer of edges is either an identity layer or a colliding layer.*

To see it, note that if we think of the vertices in each layer $\{v_{i,1}, \ldots, v_{i,w}\}$ as written from top to bottom according to β_v, then in a locally monotone program for any vertex v the 1-edge leads to the same vertex or to a vertex below the one that follows the 0-edge. Thus, assuming both E_i^1 and E_i^0 form a matching, the only way this could happen is if they both form the same matching.

Lemma 2.1 allows to show that under random restriction, with high probability, the width decreases from w to $w-1$ in most of the layers.

We seek to construct pseudorandom generators fooling unordered constant-width branching programs. Unlike the coin-problem, given a candidate for a pseudorandom generator, i.e., given some distribution D on $\{0,1\}^n$, it is not necessarily true that the best distinguisher between D and the uniform distribution is a locally monotone branching program. Thus, it is not clear that local-monotonicity would be helpful in the most general setting. Nonetheless, it does not rule out its usefulness to analyze specific candidate distributions, nor to bound complexity measures that arise in such an analysis. In particular, we will bound $L_{1,k}(B) := \sum_{s:|s|=k} |\widehat{B}(s)|$ by using bounds on locally monotone branching programs. Unlike the case $k = 1$, it is not necessarily the case that $L_{1,k}(B)$ increases when performing local monotonization. (For example, if B is the width-2 branching program computing the parity of the first 2-bits, then local monotonization reduces $L_{1,2}(B)$ from 1 to 0). We will use the structure of the branching program (namely Eq. (1)) to bound $L_{1,k}(B)$ by the sum of products of Fourier coefficients of smaller sets in sub-programs of B. This idea was considered in the works of [RSV13] and [SVW14], however the natural calculation incurs a multiplicative factor of nw going from $L_{1,k}$ to $L_{1,k+1}$, which results in trivial bounds already for $k = 2$.[3] In the following section, we describe how to avoid incurring such a large multiplicative factor.

2.2 Proof Overview.

We focus on the case $k = 2$, since it captures all the ideas in the proof. For any $i \in [n]$, using Eq. (1) we express B as a sum of products of sub-branching programs of length $i-1$ and sub-branching programs of length $n-(i-1)$, namely $B(x) = \sum_{v\in V_i} B_{\to v}(x)\cdot B_{v\to}(x)$. This means that the Fourier coefficient of a set $s \subseteq [n]$ equals $\widehat{B}(s) = \sum_{v\in V_i} \widehat{B_{\to v}}(s\cap[i-1])\cdot\widehat{B_{v\to}}(s\cap[i,n])$. In particular, if $s = \{j, i\}$ where $j < i$, then

$$\widehat{B}(\{j,i\}) = \sum_{v\in V_i} \widehat{B_{\to v}}(\{j\}) \cdot \widehat{B_{v\to}}(\{i\}) \ ,$$

Thus, we can write

$$L_{1,2}(B) = \sum_{i,j:j<i} |\widehat{B}(\{j,i\})| = \sum_{i,j:j<i} \left|\sum_{v\in V_i} \widehat{B_{\to v}}(\{j\}) \cdot \widehat{B_{v\to}}(\{i\})\right| \le \sum_{i=1}^{n}\sum_{v\in V_i} L_{1,1}(B_{\to v}) \cdot \left|\widehat{B_{v\to}}(\{i\})\right| .$$

We may use the universal upper bound given by [Ste13, SVW14] on $L_{1,1}(\cdot)$ for all branching programs of length at most n and width at most w: $L_{1,1}(B_{\to v}) \le O(\log n)^{w-2}$. Then, we would get

$$L_{1,2}(B) \le O(\log n)^{w-2} \cdot \sum_{i=1}^{n}\sum_{v\in V_i} \left|\widehat{B_{v\to}}(\{i\})\right| ,$$

and all is left is to bound the quantity $\sum_{i=1}^{n}\sum_{v\in V_i} \left|\widehat{B_{v\to}}(\{i\})\right| =: S$. Trivially, $S \le n \cdot w$, since every Fourier coefficient is at most 1 in absolute value. One might hope to get a better bound on S, perhaps poly-logarithmic in n, however for the branching program of width-3 computing the Tribes function, we have $S = \Theta(n/\log n)$. Thus, this calculation cannot guarantee something better than $L_{1,2}(B) \le O(\log n)^{w-2} \cdot \Theta(n/\log n)$, which is trivial.[4]

To avoid this, when bounding $L_{1,1}(B_{\to v})$ we take into account the probability that $B_{\to v}(x) = 1$ (i.e., the probability of reaching v). Suppose there exists a $t = t(n, w)$ such that

$$\sum_{i=1}^{n} |\widehat{M}(\{i\})| \le t \cdot \mathbf{Pr}[M(x) = 1] \qquad (2)$$

[3] a different recurrence (suggested in [RSV13], that we use in Sections 6 and 7) shows that $L_{1,2k} \le (L_{1,k})^2 \cdot n$, making it sufficient to prove a bound of the form $L_{1,k} \le O(t)^k$ only for $k \le \log n$ in order to deduce a similar bound for all $k \in [n]$.

[4] since for any Boolean function f, we have $L_{1,2}(f) \le \sqrt{\binom{n}{2} \cdot \sum_{|s|=2} \widehat{f}(s)^2} \le \sqrt{\binom{n}{2}} < n$, by Cauchy-Schwarz Inequality and Parseval Identity.

for all branching programs M of width at most w and length at most n. Then, using our assumption on $B_{\to v}$ we would get

$$L_{1,2}(B) = \sum_{i=1}^{n} \sum_{v \in V_i} L_{1,1}(B_{\to v}) \cdot \left|\widehat{B_{v\to}}(\{i\})\right| \le \sum_{i=1}^{n} \sum_{v \in V_i} t \cdot \mathbf{Pr}[B_{\to v}(x) = 1] \cdot \left|\widehat{B_{v\to}}(\{i\})\right| .$$

To bound $\sum_{i=1}^{n} \sum_{v \in V_i} \mathbf{Pr}[B_{\to v}(x) = 1] \cdot \left|\widehat{B_{v\to}}(\{i\})\right|$ we use local monotonization. Let B' be the local monotonization of B. We observed that this transformation does not change the probabilities of reaching any specific vertex in the program. Moreover, in the resulting program $\widehat{B'_{v\to}}(\{i\}) = \left|\widehat{B_{v\to}}(\{i\})\right|$ for any $v \in V$. Thus, we get

$$\sum_{i=1}^{n} \sum_{v \in V_i} \mathbf{Pr}[B_{\to v}(x) = 1] \cdot |\widehat{B_{v\to}}(\{i\})| = \sum_{i=1}^{n} \sum_{v \in V_i} \mathbf{Pr}[B'_{\to v}(x) = 1] \cdot \widehat{B'_{v\to}}(\{i\}) = \sum_{i=1}^{n} \widehat{B'}(\{i\})$$

where we use the decomposition given in Eq. (1) for B' in the last equality. Using the bound on $\sum_i |\widehat{B'}(\{i\})| \le t \cdot \mathbf{Pr}[B'(x) = 1]$ (Eq. (2)) and using $\mathbf{Pr}[B(x) = 1] = \mathbf{Pr}[B'(x) = 1]$ we get

$$L_{1,2}(B) \le t \cdot t \cdot \mathbf{Pr}[B'(x) = 1] = t^2 \cdot \mathbf{Pr}[B(x) = 1] .$$

Overall, from a bound of the form $L_{1,1}(B) \le t \cdot \mathbf{Pr}[B(x) = 1]$ for all branching programs of length at most n and width at most w, we deduce the bound $L_{1,2}(B) \le t^2 \cdot \mathbf{Pr}[B(x) = 1]$ for all such programs. A simple induction shows that we can prove for any $k \in [n]$ that $L_{1,k}(B) \le t^k \cdot \mathbf{Pr}[B(x) = 1]$.

This is all nice and elegant, however we still need to prove that there exists a small t (hopefully poly-logarithmic) for which Eq. (2) holds. The problematic examples for deriving such a bound with small t are programs for which $\mathbf{Pr}[B(x) = 1]$ is extremely small. For example, when B is the AND of n variables we have $p = \mathbf{Pr}[B(x) = 1] = 2^{-n}$ and $\sum_i \widehat{B}(\{i\}) = p \cdot n$ showing that in general t cannot be smaller than n.

It turns out that if we do not insist on a bound of the form $t \cdot \mathbf{Pr}[B(x) = 1]$ but rather $t \cdot \mathbf{Pr}[B(x) = 1] \cdot \log(1/\mathbf{Pr}[B(x) = 1])$, then we can indeed prove such a result with poly-logarithmic t. More precisely, following Steinberger's techniques [Ste13] we show

$$\sum_i |\widehat{B}(\{i\})| \le O(\log n)^{w-2} \cdot \mathbf{Pr}[B(x) = 1] \cdot O(\log(1/\mathbf{Pr}[B(x) = 1])) .$$

To avoid incurring a large multiplicative factor of $\log(1/\mathbf{Pr}[B_{\to v}(x) = 1])$ with each iteration of our inductive argument, we restrict our attention to programs where all nodes have $p_v \ge 1/\mathrm{poly}(n)$. We call such programs as programs with no negligible vertices. For such programs, we have the bound $\sum_i |\widehat{B}(\{i\})| \le t \cdot \mathbf{Pr}[B(x) = 1]$ with $t = O(\log n)^{w-1}$ and the aforementioned proof strategy works.

In section 5, we show that any pseudorandom generator fooling branching programs with no negligible vertices also fools general branching programs of the same width and length (with only a slight increase in the error). This, together with a careful analysis of the argument suggested by Reingold et al. [RSV13] and Steinke et al. [SVW14] (the careful analysis is needed since this subclass is not closed under restrictions and sub-programs, which is required if one wants to apply their analysis in a black-box fashion) gives a construction of our pseudorandom generator with seed-length $O(\log n)^{w+1} \cdot \log\log n$.

2.3 Proving the Conjecture of Reingold, Steinke and Vadhan

Although we only need the Fourier bounds estimates for ROBPs with no negligible vertices for the analysis of the PRG, we prove a bound on the Fourier spectrum of all ROBPs regardless of whether or not they have negligible vertices. These bounds are slightly worse than the bounds we have in Lemma 4.5 and would result in a larger seed-length (by a multiplicative factor of $O(\log n)$) in the analysis of the PRG. Nevertheless, we find the fact that all ROBPs have such bounds on their Fourier spectrum to be very interesting in its own right. In particular, Theorem 2 (whose proof is given in Section 7) proves the main conjecture in the work of Reingold, Steinke and Vadhan ([RSV13, Conjecture 8.1]), namely that $\sum_{s:|s|=k} \left|\widehat{B}(s)\right| \leq O(\log n)^{wk}$ for every branching program B with length-n and width-w and every $k \in [n]$.

3 Preliminaries

Denote by U_n the uniform distribution over $\{0,1\}^n$, and by U_S for $S \subseteq [n]$ to be the uniform distribution over $\{0,1\}^S$. Denote by log and ln the logarithms in bases 2 and e, respectively.

3.1 Restrictions

Definition 3.1 (Restriction). *Let $f : \{0,1\}^n \to \mathbb{R}$ be a function. A* restriction *is a pair (J, z) where $J \subseteq [n]$ and $z \in \{0,1\}^{\bar{J}}$. We denote by $f_{J|z} : \{0,1\}^n \to \mathbb{R}$ the function f restricted according to (J, z), defined by*

$$f_{J|z}(x) = f(y), \quad \textit{where} \quad y_i = \begin{cases} x_i, & i \in J \\ z_i, & \textit{otherwise} \end{cases}.$$

Definition 3.2 (Random Valued Restriction). *Let $n \in \mathbb{N}$. A random variable (J, z), distributed over restrictions of $\{0,1\}^n$ is called* random-valued *if conditioned on J, the variable z is uniformly distributed over $\{0,1\}^{\bar{J}}$.*

3.2 Fourier Analysis of Boolean Functions

Any function $f : \{0,1\}^n \to \mathbb{R}$ has a unique Fourier representation:

$$f(x) = \sum_{S \subseteq [n]} \widehat{f}(S) \cdot (-1)^{\sum_{i \in S} x_i},$$

where the coefficients $\widehat{f}(S) \in \mathbb{R}$ are given by $\widehat{f}(S) = \mathbf{E}_x[f(x) \cdot (-1)^{\sum_{i \in S} x_i}]$. Parseval's identity states that

$$\sum_S \widehat{f}(S)^2 = \mathop{\mathbf{E}}_x[f(x)^2].$$

In addition, it is easy to see by the formula of $\widehat{f}(S)$ that

$$|\widehat{f}(S)| \leq \mathop{\mathbf{E}}_x[|f(x)|]. \tag{3}$$

(where the latter equals $\mathbf{Pr}[f(x) = 1]$ in the case when f is Boolean, i.e., $f : \{0,1\}^n \to \{0,1\}$.) For $i \in [n]$, we write $\widehat{f}(i) := \widehat{f}(\{i\})$ for shorthand.

The following fact relates the Fourier coefficients of a Boolean function and its restriction.

Fact 3.3 (Proposition 4.17, [O'D14]). *Let $f : \{0,1\}^n \to \mathbb{R}$, let $S \subseteq [n]$, and let D be a distribution of random valued restrictions. Then,*

$$\mathop{\mathbf{E}}_{(J,z)\sim D}\left[\widehat{f_{J|z}}(S)\right] = \widehat{f}(S) \cdot \mathop{\mathbf{Pr}}_{(J,z)\sim D}[S \subseteq J]$$

We include the proof of this simple fact for completeness.

Proof. Let $(J, z) \sim D$. Then, by definition of random valued restriction, given J we have that z is a random string in $\{0,1\}^{\bar{J}}$. Fix J, and rewrite f's Fourier expansion by splitting the variables to $(J, \bar{J})$.

$$f(x) = \sum_{S\subseteq[n]} \widehat{f}(S) \cdot (-1)^{\sum_{i\in S} x_i} = \sum_{T\subseteq J} (-1)^{\sum_{i\in T} x_i} \cdot \sum_{T'\subseteq \bar{J}} \widehat{f}(T \cup T') \cdot (-1)^{\sum_{j\in T'} x_j}$$

Hence,

$$f_{J,z}(x) = \sum_{T\subseteq J} (-1)^{\sum_{i\in T} x_i} \cdot \sum_{T'\subseteq \bar{J}} \widehat{f}(T \cup T') \cdot (-1)^{\sum_{j\in T'} z_j} .$$

So the S-Fourier coefficient of $f_{J,z}$ is 0 if $S \not\subseteq J$ and it is $\sum_{T'\subseteq \bar{J}} \widehat{f}(S \cup T') \cdot (-1)^{\sum_{j\in T'} z_j}$ otherwise. In other words,

$$\widehat{f_{J,z}}(S) = \mathbb{1}_{\{S\subseteq J\}} \cdot \sum_{T'\subseteq \bar{J}} \widehat{f}(S \cup T') \cdot (-1)^{\sum_{j\in T'} z_j} ,$$

and its expectation conditioned on the event $(S \subseteq J)$ is $\widehat{f}(S)$. □

4 Fourier Growth of Bounded-Width Branching Programs

Lemma 4.1. *Let $B : \{0,1\}^n \to \{0,1\}$ be a length-n ROBP. Then,*

$$\widehat{B}(i) = \sum_{v\in V_i} \widehat{B_{v\to}}(i) \cdot p_v .$$

Proof. Using Eq. (1) we have

$$\begin{aligned}\widehat{B}(i) &= \mathop{\mathbf{E}}_{x\sim U_n}\left[\sum_{v\in V_i} B_{\to v}(x) \cdot B_{v\to}(x)(-1)^{x_i}\right] = \sum_{v\in V_i} \mathop{\mathbf{E}}_{x\sim U_n}[B_{\to v}(x) \cdot B_{v\to}(x)(-1)^{x_i}] \\ &= \sum_{v\in V_i} \mathop{\mathbf{E}}_{x\sim U_n}[B_{v\to}(x)(-1)^{x_i} | B_{\to v}(x) = 1] \cdot p_v = \sum_{v\in V_i} \mathop{\mathbf{E}}_{x\sim U_n}[B_{v\to}(x)(-1)^{x_i}] \cdot p_v,\end{aligned}$$

where the last equality follows from the fact that for uniformly drawn x, $B_{\to v}(x)$ is independent of $B_{v\to}(x)(-1)^{x_i}$. □

The following two lemmas are similar to two lemmas given in the work of Steinberger [Ste13]. They bound the total effect of bounded-width ROBPs. The first lemma bounds the total effect of width-2 programs, while the second lemma bounds the total effect of programs with any width $w \le \log n/\log\log n$. The main difference between these lemmas and the ones in the work of Steinberger [Ste13] is that we get bounds that are relative to the probability of acceptance. This plays later a crucial role as it allows induction on k when proving bounds on the sum of Fourier coefficients of size k of ROBPs.

Lemma 4.2 (The total effect of width-2 programs)**.** *Let $B : \{0,1\}^n \to \{0,1\}$ be a length-n, width-2 ROBP. Let $p := \mathbf{Pr}[B(x) = 1]$. Then,*

$$\sum_{i\in[n]} |\widehat{B}(i)| \le p \cdot \lceil \log(2/p) \rceil \ .$$

The lemma is essentially tight for the AND of k variables, denoted B_k, for any $k \le n$. Indeed, B_k can be computed by a width-2 ROBP and has $\sum_{i\in[n]} |\widehat{B_k}(i)| = k\cdot 2^{-k} = p\cdot\log(1/p)$ for $p := \mathbf{Pr}[B_k(x) = 1]$.

Proof. We assume without loss of generality that the program is locally monotone since making an ROBP locally monotone may only increase the sum $\sum_{i\in[n]} |\widehat{B}(i)|$.

We assume without loss of generality that B has no identity layers. Then, by Lemma 2.1 each layer in B has a collision. This means that for every $i \in [n]$, under a random assignment to $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$ the probability that x_i is sensitive in B is at most $2^{-(n-i)}$. We get that $\mathrm{Inf}_i(B) \le 2^{-(n-i)}$, hence

$$|\widehat{B}(i)| = \tfrac{1}{2} \cdot \mathrm{Eff}_i(B) \le \tfrac{1}{2} \cdot \mathrm{Inf}_i(B) \le \tfrac{1}{2} \cdot 2^{-(n-i)}. \tag{4}$$

We break the sum $\sum_{i=1}^n |\widehat{B}(i)|$ into the first $n - \lceil\log(1/p)\rceil$ layers and the last $\lceil\log(1/p)\rceil$ layers. For the first $n - \lceil\log(1/p)\rceil$ layers we use Eq. (4):

$$\sum_{i=1}^{n-\lceil\log(1/p)\rceil} |\widehat{B}(i)| \le \sum_{i=1}^{n-\lceil\log(1/p)\rceil} \tfrac{1}{2} \cdot 2^{-(n-i)} \le p$$

For the last $\lceil\log(1/p)\rceil$ layers, we use Eq. (3) to get $|\widehat{B}(i)| \le \mathbf{Pr}[B(x) = 1]$, thus

$$\sum_{i=n-\lceil\log(1/p)\rceil+1}^{n} |\widehat{B}(i)| \le \lceil\log(1/p)\rceil \cdot \mathbf{Pr}[B(x) = 1] = \lceil\log(1/p)\rceil \cdot p$$

Overall, we get $\sum_{i\in[n]} \widehat{B}(i) \le p + p\cdot\lceil\log(1/p)\rceil = p\cdot\lceil\log(2/p)\rceil$. □

Next, we prove that the total-influence of a width-w ROBP B with $p = \mathbf{Pr}[B(x) = 1]$ is at most $O(\log n)^{w-2} \cdot p \cdot \log(4/p)$. Our upper bound is tight for $w = 3$ as demonstrated by the negation of a tribe-like function with $\ln(1/p)\cdot 2^w$ tribes of width w each, such that $n = w \cdot 2^w \cdot \ln(1/p)$. Such a function can be computed by a width-3 ROBP B for which $\mathbf{Pr}[B(x) = 1] = \Theta(p)$ and $\sum_i |\widehat{B}(i)| = \Theta(p\cdot\log(1/p)\cdot\log(n))$.

Lemma 4.3 (The total effect of width-w programs)**.** *There exists a universal constant $C > 0$ such that the following holds. Let $B : \{0,1\}^n \to \{0,1\}$ be a length-n, width-w ROBP. Let $p = \mathbf{Pr}[B(x) = 1]$. Then,*

$$\sum_{i\in[n]} |\widehat{B}(i)| \le (C \cdot \log n)^{w-2} \cdot p \cdot \log(4/p)\ .$$

Proof. We prove the lemma by an induction on w. The base case with $w = 2$ follows from Lemma 4.2. For the inductive step, we assume that the lemma holds for width w, and prove it for any branching program B of width $w + 1$.

Let B be a branching program of width $w + 1$. We assume without loss of generality that B has no identity layers. We also assume without loss of generality that the program is locally monotone, since making an ROBP locally monotone may only increase the sum $\sum_{i\in[n]} |\widehat{B}(i)|$. For a locally monotone program, all $\widehat{B}(i)$ are non-negative so it is enough to bound $\sum_{i\in[n]} \widehat{B}(i)$.

Let $a := \lceil \log(2n^2)\rceil$. We apply Steinberger's argument [Ste13]. We hit the function with a random restriction that keeps exactly one out of every a layers alive. The restriction may depend on the function itself, as it depends on the order that the bits are read. This is a non-standard random restriction procedure that takes into account the structure of the BP it is applied to. Nevertheless, we shall see that it still behaves well with respect to the total effect of the program.

We pick $j \in_R \{0, \dots, a-1\}$ uniformly at random, and keep alive every layer whose index is j modulo a. That is, the set $J = \{i \in [n] : i \equiv j \mod a\}$ is the set of alive variables (see Definition 3.1). We randomly assign all other layers with uniform random bits.

Denote by B' the random variable describing the residual branching program after the random restriction. The analysis is based on two ideas:

1. The expected $\sum_{i=1}^{n} \widehat{B'}(i)$ equals $\frac{1}{a} \cdot \sum_{i=1}^{n} \widehat{B}(i)$.
2. For all $b \in \mathbb{N}$, with probability at least $1 - n^b$ (over the random restriction), all but at most b alive layers in B' have at most w reachable nodes in them.

We begin with the first item. By Fact 3.3, the expected value of $\widehat{B'}(i)$ is exactly the probability that x_i remains alive times $\widehat{B}(i)$. The marginal probability that each x_i remains alive is $1/a$, so we get

$$\mathbf{E}\left[\sum_i \widehat{B'}(i)\right] = \sum_i \widehat{B}(i) \cdot \mathbf{Pr}[x_i \text{ remains alive}] = \frac{1}{a} \cdot \sum_i \widehat{B}(i).$$

Next, we show the second item. By our assumption that B has no identity layers, and using Lemma 2.1, every layer in B is colliding – meaning that either the 0-edges or the 1-edges have a collision. When picking values to a block of $a-1$ consecutive layers of edges uniformly at random, the probability that a collision would not happen is at most $1/2^{a-1}$. When such a collision occurs, in the layer of vertices that follow this block (of $a - 1$ layers of edges), at most w out of the $w + 1$ vertices are reachable in the restricted branching program. In the layer of vertices that follow the first block of fixed layers, which has potentially less than

$a-1$ layers of edges, only one vertex is reachable. If a collision occured in all blocks, then the restricted branching program is equivalent to a width-w branching program. Consider the n/a events corresponding to whether or not a collision occurred in the different blocks. Once j is fixed, these n/a events are independent. Let $b \in \mathbb{N}$. By union bound over all subsets of blocks of size b, the probability that there are at least b "bad blocks" is at most $\binom{n/a}{b} \cdot (1/2^{(a-1)})^b \le n^b \cdot n^{-2b} = n^{-b}$. This completes the proof of the second item.

We turn to prove the bound on $\sum_{i=1}^n \widehat{B}(i)$. By the first item, we have

$$\sum_{i=1}^{n} \widehat{B}(i) \le a \cdot \mathbf{E}\left[\sum_{i=1}^{n} \widehat{B'}(i)\right].$$

Let $b := \lceil 2 + \log(1/p) \rceil$. We use the second item to bound the expected total effect of B'. The case where B' has more than b bad blocks has probability at most $n^{-b} \le p/n^2$. In such a case the total effect of B' is at most n so overall, this case contributes at most $a \cdot p/n^2 \cdot n \le p$ to the total effect of B.

In the case where B' has at most b bad blocks we have the following upper bound on $\sum_i \widehat{B'}(i)$. First denote by $I \subseteq [n]$ the set of alive variables who correspond to alive layers of edge that immediately precede a bad block. Fix the variables with indices in I to uniformly random values. We get a new (random variable) branching program, B'', that is equivalent to a branching program with width at most w. Thus, by the induction hypothesis, the total effect of B'' is at most

$$\sum_i |\widehat{B''}(i)| \le (C \cdot \log n)^{w-2} \cdot \mathbf{Pr}[B''(x) = 1] \cdot \log(4/\mathbf{Pr}[B''(x) = 1]).$$

Using the fact that every Fourier coefficient of B' is at most $\mathbf{Pr}[B'(x) = 1]$ (see Eq. (3)), and Fact 3.3, we get that B' has

$$\begin{aligned}\sum_i \widehat{B'}(i) &= \sum_{i \in I} \widehat{B'}(i) + \mathop{\mathbf{E}}_{B''}\left[\sum_{i \notin I} \widehat{B''}(i)\right] \\ &\le b \cdot \mathbf{Pr}[B'(x) = 1] + (C \cdot \log n)^{w-2} \cdot \mathop{\mathbf{E}}_{B''}[\mathbf{Pr}[B''(x) = 1] \cdot \log(4/\mathbf{Pr}[B''(x) = 1])]\end{aligned}$$

We use the facts that $x \cdot \log(4/x)$ is concave for $x \in [0,1]$ and that $\mathbf{E}_{B''}[\mathbf{Pr}[B''(x) = 1]] = \mathbf{Pr}[B'(x) = 1]$ to get

$$\begin{aligned}\mathop{\mathbf{E}}_{B''}[\mathbf{Pr}[B''(x) = 1] \cdot \log(4/\mathbf{Pr}[B''(x) = 1])] &\le \mathop{\mathbf{E}}_{B''}[\mathbf{Pr}[B''(x) = 1]] \cdot \log(4/\mathop{\mathbf{E}}_{B''}[\mathbf{Pr}[B''(x) = 1]]) \\ &= \mathbf{Pr}[B'(x) = 1]] \cdot \log(4/\mathbf{Pr}[B'(x) = 1]]) .\end{aligned}$$

This gives the bound

$$\sum_i \widehat{B'}(i) \le b \cdot \mathbf{Pr}[B'(x) = 1] + (C \cdot \log n)^{w-2} \cdot \mathbf{Pr}[B'(x) = 1] \cdot \log(4/\mathbf{Pr}[B'(x) = 1]) .$$

We return to bound the total effect of B.

$$\sum_i \widehat{B}(i) = a \cdot \mathop{\mathbf{E}}_{B'}\left[\sum_i \widehat{B'}(i)\right] \le p + a \cdot \mathop{\mathbf{E}}_{B'}\left[\sum_i \widehat{B'}(i) \cdot \mathbb{1}_{\{B' \text{ has at most } b \text{ bad layers}\}}\right]$$

$$\leq p + a \cdot \mathop{\mathbf{E}}_{B'}[b \cdot \mathbf{Pr}[B'(x)=1]] + a \cdot \mathop{\mathbf{E}}_{B'}\left[(C \cdot \log n)^{w-2} \cdot \mathbf{Pr}[B'(x)=1] \cdot \log(4/\,\mathbf{Pr}[B'(x)=1])\right] .$$

Using the concavity of $x \cdot \log(4/x)$ again and using $\mathbf{E}_{B'}[\mathbf{Pr}[B'(x)=1]] = \mathbf{Pr}[B(x)=1]$ gives

$$\sum_i \widehat{B}(i) \leq p + a \cdot b \cdot p + a \cdot (C \cdot \log n)^{w-2} \cdot p \log(4/p) \leq (C \cdot \log n)^{w-1} \cdot p \log(4/p) ,$$

where the last inequality holds by the choices of $a = \lceil \log(2n^2) \rceil$, $b = \lceil 2 + \log(1/p) \rceil$ and for a large enough constant $C > 0$. This completes the induction, hence the lemma follows. □

Combining Lemma 4.1 and Lemma 4.3 we get:

Corollary 4.4. *There exists a universal constant $C > 0$ such that the following holds. Let B be a length-n, width-w ROBP. Denote by $p = \mathbf{Pr}[B(x)=1]$. Then,*

$$\sum_{i \in [n], v \in V_i} |\widehat{B_{v\to}}(i)| \cdot p_v \leq (C \cdot \log n)^{w-2} \cdot p \cdot \log(4/p) .$$

Proof. We wish to upper bound $\sum_{i \in [n], v \in V_i} |\widehat{B_{v\to}}(i)| \cdot p_v$. First, we locally monotonize the program. By relabeling the edges we generate the program B' with

$$\widehat{B'_{v\to}}(i) = \left|\widehat{B_{v\to}}(i)\right|$$

for all i. Denote by $p = \mathbf{Pr}[B(x)=1]$, by $p' = \mathbf{Pr}[B'(x)=1]$, by $p_v = \mathbf{Pr}[B_{\to v}(x)=1]$ and by $p'_v = \mathbf{Pr}[B'_{\to v}(x)=1]$. Since the probability of reaching any state remains the same under the relabeling we get $p' = p$ and $p'_v = p_v$ for any vertex v. Using Lemma 4.3 and Lemma 4.1, we conclude that

$$\begin{aligned} \sum_{i \in [n], v \in V_i} |\widehat{B_{v\to}}(i)| \cdot p_v &= \sum_{i \in [n], v \in V_i} \widehat{B'_{v\to}}(i) \cdot p'_v = \sum_{i \in [n]} \widehat{B'}(i) && \text{(Lemma 4.1)} \\ &\leq (C \cdot \log n)^{w-2} \cdot p' \cdot \log(4/p') && \text{(Lemma 4.3)} \\ &= (C \cdot \log n)^{w-2} \cdot p \cdot \log(4/p) . \end{aligned}$$ □

Lemma 4.5. *Let B be a length-n, width-w ROBP with $w \leq n$. Suppose for every vertex v in B it holds that $p_v \geq \delta > 0$. Then*

$$\sum_{s:|s|=k} \left|\widehat{B}(s)\right| \leq \left(O(\log n)^{w-2} \cdot \log(4/\delta)\right)^k \cdot \mathbf{Pr}[B(x)=1]$$

for all $k \in [n]$.

This proves [RSV13, Conjecture 8.1] in the case where all vertices in the branching programs have $p_v \geq 1/\mathrm{poly}(nw)$. In Section 5, we show that it is enough to fool such programs, since any PRG that fools branching programs with $p_v \geq 1/\mathrm{poly}(nw)$ also fools general branching programs with comparable error. Then, in Section 6 we show that the iterated random restriction generator suggested by Gopalan et al. [GMR+12], with seed-length $O(\log n)^{w+1}$, $1/\mathrm{poly}(n)$-fools general length-n width-w ROBPs. The latter argument is more complicated than we thought initially, because the subclass of programs with no negligible vertices is not closed under restrictions and sub-programs (and the black-box reduction stated in [SVW14] requires such a property from the subclass of BPs). Nevertheless, we manage to bypass this difficulty in Section 6.

Proof. Denote by $t = (C \cdot \log n)^{w-2} \cdot \log(4/\delta)$ for the constant $C > 0$ from the statement of Corollary 4.4. We prove, by induction on $k \in \mathbb{N}$, that for all ROBP B of length at most n width at most w such that $p_v \geq \delta$ for all vertices v in the BP, it holds that

$$\sum_{s:|s|=k} \left|\widehat{B}(s)\right| \leq t^k \cdot \mathbf{Pr}[B(x) = 1].$$

The base case is already given by Lemma 4.3. Assuming the claim holds for k, we show that it holds for $k + 1$. Recall the definitions of $B_{\to v}$, $B_{v\to}$ and p_v. Using Equation (1) we have

$$\begin{aligned}
\sum_{s:|s|=k+1} \left|\widehat{B}(s)\right| &\leq \sum_{i\in[n]} \sum_{\substack{|s'|=k,\\ v\in V_i}} \left|\widehat{B_{\to v}}(s')\right| \cdot \left|\widehat{B_{v\to}}(i)\right| \\
&\leq \sum_{i\in[n]} \sum_{v\in V_i} |\widehat{B_{v\to}}(i)| \cdot \sum_{|s'|=k} \left|\widehat{B_{\to v}}(s')\right| \\
&\leq \sum_{i\in[n]} \sum_{v\in V_i} |\widehat{B_{v\to}}(i)| \cdot \left(t^k \cdot \mathbf{Pr}[B_{\to v}(x) = 1]\right) \\
&\qquad\qquad \text{(induction on } B_{\to v} \text{ and sets of size } k\text{)} \\
&\leq t^k \cdot \sum_{i\in[n]} \sum_{v\in V_i} |\widehat{B_{v\to}}(i)| \cdot p_v \qquad \text{(since } p_v = \mathbf{Pr}[B_{\to v}(x) = 1]\text{)} \\
&\leq t^k \cdot (C \cdot \log n)^{w-2} \cdot \mathbf{Pr}[B(x) = 1] \cdot \log\left(\tfrac{4}{\mathbf{Pr}[B(x)=1]}\right) \qquad \text{(Corollary 4.4)} \\
&\leq t^k \cdot (C \cdot \log n)^{w-2} \cdot \mathbf{Pr}[B(x) = 1] \cdot \log(4/\delta) \qquad \text{(since } \mathbf{Pr}[B(x) = 1] \geq \delta\text{)} \\
&\leq t^{k+1} \cdot \mathbf{Pr}[B(x) = 1] \ .
\end{aligned}$$

□

5 PRGs for branching programs with no negligible vertices is enough

Notation. We call a read-once branching program B an (n, w, δ)-ROBP if B is of length-n, width-w and for all reachable vertices v in B we have $p_v(B) \geq \delta$ where

$$p_v(B) := \mathop{\mathbf{Pr}}_{x\sim U_n} [\text{reaching } v \text{ on the walk on } B \text{ defined by } x].$$

Theorem 5.1. *Let D be a distribution on $\{0, 1\}^n$ that ε-fools all (n, w, δ)-ROBPs. Then, D also $O((\varepsilon + \delta)nw)$-fools any $(n, w, 0)$-ROBPs.*

Proof. Let D be a distribution on $\{0, 1\}^n$ that ε-fools all (n, w, δ)-ROBPs. We show that D also $O((\varepsilon + \delta)nw)$-fools any $(n, w, 0)$-ROBPs. The first observation is that any distribution D that fools all (n, w, δ)-ROBPs also fools prefixes of these programs. The reason is simple because to simulate the prefix of length-k of a (n, w, δ)-program B, one can simply reroute the last $n - k$ layers of edges in B so that they would "do nothing", i.e. that they would be the identity transformation regardless of the values of $x_{k+1}, \ldots, x_n$.

Let B be a length n width-w ROBP. Next, we introduce B', an (n, w, δ)-ROBP, that would help bound the difference between

$$B(U_n) := \mathop{\mathbf{Pr}}_{x\sim U_n} [B(x) = 1] \quad \text{and} \quad B(D) := \mathop{\mathbf{Pr}}_{x\sim D} [B(x) = 1] \ ,$$

where U_n is the uniform distribution over $\{0,1\}^n$. Let B' be the the following modified version of B. To construct B' we consider a sequence of $n+1$ branching programs $B_0,\dots,B_n$ where $B_0 = B$ and $B' = B_n$. We initiate B_0 with B. For $i = 1,\dots,n$ we take B_i to be B_{i-1} except we may reroute some of the edges in the i-th layer (of edges). We explain the rerouting procedure. For $i = 1,\dots,n$ we calculate the probability to reach vertices in layer V_i of B_{i-1}. If some vertex v in the i-th layer has probability less than 2δ, then we reroute the two edges going from the vertex v to go to the same vertex in the $(i+1)$-th layer (the choice of this vertex may be arbitrary, so we pick the first vertex according to some canonical order). We denote by V_{small} the set of vertices for which we rerouted the outgoing edges from them.

First, we claim that any reachable vertex v in B_n has $p_v \geq \delta$. We apply induction and show that for $i = 0,\dots,n$ any vertex reachable by B_i in layers $1,\dots,i+1$ has $p_v \geq \delta$. This obviously hold for the starting vertex in B_0 which has $p_v = 1$. To apply induction assume the claim holds for B_{i-1} and show that it holds for B_i. The claim obviously holds for all vertices in layers $0,1,\dots,i$ in B_i since we didn't change any edge in those layers going from B_{i-1} to B_i. Let v be a reachable vertex in the $(i+1)$-th layer of B_i. It means that there is a vertex in v' in the i-th layer of B_i (and also in B_{i-1}) that has an outgoing edge to v. By induction, $p_{v'}(B_{i-1}) \geq \delta$. In addition, if $p_{v'}(B_{i-1}) < 2\delta$ then both edges from v' should go to v and hence $p_v(B_i) \geq p_{v'}(B_{i-1}) \geq \delta$. In the other case $p_{v'}(B_{i-1}) \geq 2\delta$ and there's at least one edge going from v' to v, thus $p_v(B_i) \geq \frac{1}{2} \cdot p_{v'}(B_{i-1}) \geq \delta$. (Of course, there can be unreachable vertices in B_i, that were reachable in B_{i-1} but we do not account for them.)

To bound $|B(U_n) - B(D)|$ we use the triangle inequality:

$$|B(U_n) - B(D)| \leq |B(U_n) - B'(U_n)| + |B'(U_n) - B'(D)| + |B'(D) - B(D)|,$$

and bound each of the three terms separately:

1. The first term is bounded by the probability of reaching one of the nodes in V_{small} in B' when taking a uniform random walk. This follows since if the path defined by x didn't pass through V_{small} then we would end up with the same node in both B and B' (since no rerouting effected the path). By union bound, the probability to pass through V_{small} is at most $|V_{\text{small}}| \cdot 2\delta$.

2. The second term is at most ε since the program B' has all $p_v \geq \delta$.

3. Similarly to the first term, the third term is bounded by the probability of reaching one of the nodes in V_{small} in B' when taking a walk sampled by D.

$$\begin{aligned}|B'(D) - B(D)| &\leq \mathop{\mathbf{Pr}}_{x\sim D}[\text{reaching } V_{\text{small}} \text{ on the walk on } B' \text{ defined by } x] \\ &\leq \sum_{v\in V_{\text{small}}} \mathop{\mathbf{Pr}}_{x\sim D}[\text{reaching } v \text{ on the walk on } B' \text{ defined by } x]\end{aligned}$$

However since D is pseudorandom for prefixes of B', for each $v \in V_{\text{small}}$ the probability of reaching v when walking according to D is ε-close to the probability of reaching v when walking according to U_n.

$$|B'(D) - B(D)| \leq \sum_{v\in V_{\text{small}}} \mathop{\mathbf{Pr}}_{x\sim U_n}[\text{reaching } v \text{ on the walk on } B' \text{ defined by } x] + \varepsilon$$

$$= \sum_{v \in V_{\text{small}}} (p_v(B') + \varepsilon) \leq |V_{\text{small}}| \cdot (\varepsilon + 2\delta)$$

Summing the upper bound on the three terms gives:

$$|B(U_n) - B(D)| \leq |V_{\text{small}}| \cdot (\varepsilon + 4\delta) + \varepsilon \leq O(nw(\varepsilon + \delta)). \qquad \square$$

6 The Pseudorandom Generator

In this section, we will present our pseudorandom generator and its proof of correctness, proving Theorem 1. We use a pseudorandom generator that was suggested by Gopalan et al. [GMR+12] in the context of fooling read-once CNF's and combinatorial rectangles.

Our analysis is inspired by that of Reingold et al. [RSV13] and Steinke et al. [SVW14], however we need a few extra ideas as the class of branching programs without negligible vertices is not closed under restrictions and subprograms. Before stating the lemma, we will define the kind of pseudorandom restrictions that we will utilize.

Definition 6.1 (Almost (p,k)-wise independence). *A random variable $X \in \{0,1\}^n$ is said to be δ-almost (p,k)-wise independent, for $\delta, p \in [0,1]$ and $k \in \mathbb{N}$, if for any subset $I \subseteq [n]$ of size at most k, and any assignment $z \in \{0,1\}^I$ with $n_0 := |\{i \in I : z_i = 0\}|$ zeros and $n_1 := |\{i \in I : z_i = 1\}|$ ones, it holds that*

$$\left| \mathop{\mathbf{Pr}}_{X}[X|_I = z] - p^{n_1} \cdot (1-p)^{n_0} \right| \leq \delta\ .$$

We say that a random $S \subseteq [n]$ is δ-almost (p,k)-wise independent, if its Boolean representation $1_S \in \{0,1\}^n$ is δ-almost (p,k)-wise independent.

6.1 One step of the Pseudorandom Generator

In this section we show the analysis of one step of our generator.

Theorem 6.2. *Given n, $3 \leq w \leq \frac{\log n}{\log\log n}$ and $\varepsilon > 0$, there is a choice of $k = O(\log(n/\varepsilon))$, $p = 1/(O(\log n)^{w-2} \cdot \log(n/\varepsilon))$, $\delta = p^{2k}$ and $\sigma = O(\text{poly}(\varepsilon/n))$ such that the following holds.*

Let π be a permutation in S_n. Let B^π be an unordered ROBP of length n and width at most w that reads its input in order $x_{\pi_1}, \dots, x_{\pi_n}$. Let $T \subseteq [n]$ be chosen according to a δ-almost $(p,2k)$-wise independent distribution D. Moreover, let D_x be a σ-biased distribution over $\{0,1\}^n$. Then

$$\left| \mathop{\mathbf{E}}_{u \sim U_n}[B^\pi(u)] - \mathop{\mathbf{E}}_{\substack{T \sim D, \\ u \sim U_{\overline{T}}, x \sim D_x}}[B^\pi_{T|u}(x))] \right| \leq \frac{\varepsilon}{n}.$$

Proof. We observe that applying π to the distributions D and D_x does not affect their δ-almost $(p,2k)$-independence and σ-biasedness respectively. Hence, it suffices to prove the theorem for the ordered case, namely when π is the identity.

By Theorem 5.1, it is sufficient to prove for ROBPs with all nodes satisfying $p_v \geq \frac{\varepsilon}{O((nw)\cdot n)}$ that

$$\left| \underset{u\sim U_n}{\mathbf{E}}[B(u)] - \underset{\substack{T\sim D,\\ u\sim U_{\overline{T}}, x\sim D_x}}{\mathbf{E}}[B_{T|u}(x))] \right| \leq \frac{\varepsilon}{O(n\cdot(nw))}.$$

Thus assume $p_v \geq \rho$, where $\rho := \frac{\varepsilon}{O((nw)\cdot n)} \geq \frac{\varepsilon}{O(n^3)}$. We will prove that in absence of negligible vertices, $L_1(\mathbf{E}_{u\sim U_{\overline{T}}}[B_{T|u}]) \leq \mathrm{poly}(n/\epsilon)$ with probability greater than $1-\frac{\varepsilon}{n^3}$, and thus $\mathbf{E}_{u\sim U_{\overline{T}}}[B_{T|u}]$ is fooled by a σ-biased distribution. This follows from the next lemma and our choice of k.

Lemma 6.3.

$$\Pr_T\left[L_1(\underset{u\sim U_{\overline{T}}}{\mathbf{E}}[B_{T|u}]) \geq \Omega(\mathrm{poly}(n/\varepsilon))\right] \leq \frac{\mathrm{poly}(n)}{2^k}. \tag{5}$$

We postpone the proof of this lemma for now and see how it immediately implies Theorem 6.2. Note that there is a choice of $k = O(\log n + \log\frac{1}{\varepsilon})$ for which $\frac{\mathrm{poly}(n)}{2^k} \leq \frac{\varepsilon}{n^3}$. Let $\mathcal{E}$ be the event that the set T satisfies the condition of Lemma 6.3. We have

$$\begin{aligned}\left| \underset{u\sim U_n}{\mathbf{E}}[B(u)] - \underset{\substack{T,u\sim U_{\overline{T}},\\ x\sim D_x}}{\mathbf{E}}[B_{T|u}(x))] \right| &\leq \left| \underset{u\sim U_n}{\mathbf{E}}[B(u)] - \underset{\substack{T,u\sim U_{\overline{T}},\\ x\sim D_x}}{\mathbf{E}}[B_{T|u}(x)) \mid \bar{\mathcal{E}}\ ] \right| \cdot \Pr_T[\bar{\mathcal{E}}] + \Pr_T[\mathcal{E}] \\ &\leq \sigma\cdot\mathrm{poly}(n/\varepsilon) + O\left(\frac{\varepsilon}{n^3}\right) \leq O\left(\frac{\varepsilon}{n^3}\right).\end{aligned}$$

where the last inequality follows from Lemma 6.3 and the choices of k and σ. □

Before presenting the proof of Lemma 6.3, let us introduce some more notation. Given two nodes $v_1 \in V_i$ and $v_2 \in V_j$, for $1 \leq i \leq j \leq n+1$, denote by $B_{v_1\to v_2}$ the sub-branching program of B that starts at the node v and ends on layer j with v_2 as the accepting node.

Proof of Lemma 6.3: We will prove that with high probability a class of branching programs deduced from different single-layer relabelings of $B_{\to v}$'s have small L_1 Fourier weight on layers up to $2k$, and use this along with the structure of branching programs to bound the higher layer Fourier weights using these lower layers in an inductive manner.

Let us define what we mean by a single-layer relabelling. Let M be a branching program of length n and width w. For $i \in [n]$ and $a = (a_1, ..., a_w) \in \{0,1\}^w$, define $M^{(i,a)}$ to be the branching program deduced from M by relabelling all the edges with a starting node at layer i according to a: We will think of a_j telling us whether to switch the labels of the edges going out from the j-th node in the i-th layer (i.e., $v_{i,j}$).

We store a simple but useful fact about relabelings of a branching program.

Claim 6.4. *Let M be any branching program of length m and width w with layers $V_1, ..., V_{m+1}$. For any set $s \subseteq [m]$, and $i = min\{j : j \in s\}$, there exists $a \in \{0,1\}^w$ such that for each $v \in V_i$, $|\widehat{M_{v\to}}(s)| = \widehat{M^{(i,a)}_{v\to}}(s)$.*

Proof. Let $v \in V_i$ be the r-th node in V_i. We show how to set the r^{th} bit of the string a. Suppose the 0-labelled edge out of v is to v_0 and the 1-labelled edge out of v is to v_1 (where v_0, v_1 are nodes in V_{i+1}). We have

$$\widehat{M_{v\to}}(s) = \mathbf{E}[M_{v\to}(x)(-1)^{x_i+\sum_{j\in s\setminus\{i\}} x_j}]$$

$$= \frac{1}{2} \cdot \Big(\mathbf{E}[M_{v\rightarrow}(x)(-1)^{\sum_{j \in s\setminus\{i\}} x_j} | x_i = 1] - \mathbf{E}[M_{v\rightarrow}(x)(-1)^{\sum_{j \in s\setminus\{i\}} x_j} | x_i = 1] \Big)$$
$$= \frac{1}{2} \cdot \Big(\mathbf{E}[M_{v_0\rightarrow}(x)(-1)^{\sum_{j \in s\setminus\{i\}} x_j}] - \mathbf{E}[M_{v_1\rightarrow}(x)(-1)^{\sum_{j \in s\setminus\{i\}} x_j}]] \Big)$$

Thus, depending on the sign of $\widehat{M_{v\rightarrow}}(s)$, we can either flip the edge (which as evident from the above calculation, just flips the sign of $\widehat{M_{v\rightarrow}}(s)$) or retain the labelling such that the modified branching program M' is such that $\widehat{M'}(s) = |\widehat{M}(s)|$. This fixes the r^{th} bit of the string a. We can fix the other bits of a similarly. This completes the proof. □

We are now ready to prove that the layers up to $2k$ of the Fourier expansion have a small L_1 norm under the restriction.

Claim 6.5. *For all $\beta > 0$, the following holds with probability at least $1 - \frac{2^w \cdot w \cdot n^3}{\beta}$ over T, for all $\ell \in [n+1]$, $i \le \ell$, $a \in \{0,1\}^w$, $v \in V_\ell$ and $1 \le j \le \min\{2k, n\}$:*

$$L_{1,j}\big(\mathop{\mathbf{E}}_{u \sim u_{\overline{T}}}[(B^{(i,a)}_{\rightarrow v})_{T|u}]\big) \le \frac{\beta}{2^j}.$$

Proof. We note that if B has no negligible vertices, then same is true for $B^{(i,a)}_{\rightarrow v}$ for all choices of v, i, and a. This is true because the probability of reaching a vertex v' in $B^{(i,a)}_{\rightarrow v}$ is the same as it was in B. Fix v, i and a. Letting M denote the branching program $B^{(i,a)}_{\rightarrow v}$ and $\mathcal{M}_T := \mathbf{E}_{u \sim u_{\overline{T}}}[M_{T|u}]$, by Lemma 4.5 we get

$$\sum_{s:|s|=j} \left| \widehat{M}(s) \right| \le (O(\log n)^{w-2} \cdot O(\log(n/\varepsilon)))^j.$$

Now using Fact 3.3 we get

$$\mathop{\mathbf{E}}_{T}[L_{1,j}(\mathcal{M}_T)] = \sum_{s:|s|=j} |\widehat{M}(s)| \cdot \mathop{\mathbf{Pr}}_{T}[s \subseteq T] \le (O(\log n)^{w-2} \cdot O(\log(n/\varepsilon)))^j \cdot (p^j + \delta) \le \frac{1}{2^j}.$$

Finally, we conclude by applying the Markov inequality and a union bound, as there is a total of at most $w \cdot n^2 \cdot 2^w$ branching programs $B^{(i,a)}_{\rightarrow v}$ and at most n choices for j. □

Lemma 6.3 follows from the next claim which uses Claim 6.5 with $\beta = O(\mathrm{poly}(n/\varepsilon))$ and $k = O(\log(n/\varepsilon))$ that ensure $\frac{2^w \cdot w \cdot n^3}{\beta} \le \frac{\varepsilon}{\mathrm{poly}(n)}$ and $\frac{\beta}{2^k} \le \frac{\rho}{n \cdot 2^w}$ (recalling that $w \le \log n$).

Claim 6.6. *Suppose that T is such that the events in Claim 6.5 hold for $\beta = O(\mathrm{poly}(n/\varepsilon))$. Denote by $\mathcal{B} := \mathbf{E}_{u \sim U_{\overline{T}}}[B_T|u]$. Then for every $k \le j \le n$,*

$$\sum_{s:|s|=j} |\widehat{\mathcal{B}}(s)| \le \frac{\rho}{n \cdot 2^w}. \tag{6}$$

Proof. Using Fact 3.3, for any $s \subseteq [n]$ we have $|\widehat{\mathcal{B}}(s)| = |\widehat{B}(s)| \cdot \mathbb{1}_{\{s \subseteq T\}}$, thus Eq. (6) is equivalent to

$$\sum_{s \subseteq T, |s|=j} |\widehat{B}(s)| \le \frac{\rho}{n \cdot 2^w}. \tag{7}$$

We will prove by induction on j that Eq. (7) holds for all $B_{\to v}$, for any $\ell \in [n+1]$ and $v \in V_\ell$. Note that B itself is of the form $B_{\to v}$ for v being the accept node in the final layer (w.l.o.g. there exists only one such node). The case $k \le j \le 2k$ is handled by Claim 6.5, since $\sum_{s\subseteq T:|s|=j} |\widehat{B_{\to v}}(s)| = L_{1,j}(\mathbf{E}_{u\sim U_{\overline{T}}}[(B_{\to v})_{T|u}]) \le \frac{\beta}{2^j} \le \frac{\beta}{2^k} \le \frac{\rho}{n\cdot 2^w}$. For $j > 2k$ we have:

$$
\begin{aligned}
\sum_{s\subseteq T:|s|=j} |\widehat{B_{\to v}}(s)| &= \sum_{i\in T\cap[\ell]} \sum_{v_0\in V_i} \sum_{\substack{s_0\subseteq T\cap\{1,\dots,i-1\}:\\ |s_0|=j-k}} \sum_{\substack{s_1\subseteq T\cap\{i,\dots,\ell\}:\\ |s_1|=k, i\in s_1}} |\widehat{B_{\to v_0}}(s_0)\cdot \widehat{B_{v_0\to v}}(s_1)| \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(by Eq. (1))} \\
&\le \frac{1}{\rho} \sum_{i\in T\cap[\ell]} \sum_{v_0\in V_i} \Bigg(\sum_{\substack{s_0\subseteq T\cap\{1,\dots,i-1\}:\\ |s_0|=j-k}} |\widehat{B_{\to v_0}}(s_0)| \Bigg) \cdot \Bigg(\sum_{\substack{s_1\subseteq T\cap\{i,\dots,\ell\}:\\ |s_1|=k, i\in s_1}} p_{v_0} |\widehat{B_{v_0\to v}}(s_1)| \Bigg) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(using } p_{v_0} \ge \rho\text{)} \\
&\le \frac{\rho}{n\cdot 2^w\cdot\rho} \sum_{i\in T\cap[\ell]} \sum_{\substack{s_1\subseteq T\cap\{i,\dots,\ell\}:\\ |s_1|=k, i\in s_1}} \Bigg(\sum_{v_0\in V_i} p_{v_0} |\widehat{B_{v_0\to v}}(s_1)| \Bigg) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(by the induction hypothesis)} \\
&\le \frac{1}{n\cdot 2^w} \sum_{i\in T\cap[\ell]} \sum_{\substack{s_1\subseteq T\cap\{i,\dots,\ell\}:\\ |s_1|=k, i\in s_1}} \Bigg(\sum_{v_0\in V_i} p_{v_0} \widehat{B^{(i,a_{s_1})}_{v_0\to v}}(s_1) \Bigg) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(by Claim 6.4, there is always such an } a_{s_1}\text{)} \\
&= \frac{1}{n\cdot 2^w} \sum_{i\in T\cap[\ell]} \sum_{\substack{s_1\subseteq T:\\ |s_1|=k, i\in s_1}} \widehat{B^{(i,a_{s_1})}_{\to v}}(s_1) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(Since } p_{v_0} = \mathbf{Pr}[B \text{ reaches } v_0] = \mathbf{Pr}[B^{(i,a_{s_1})}_{\to v} \text{ reaches } v_0]\text{)} \\
&\le \frac{1}{n\cdot 2^w} \sum_{i\in T\cap[\ell]} \sum_{\substack{s_1\subseteq T:\\ |s_1|=k, i\in s_1}} \left|\widehat{B^{(i,a_{s_1})}_{\to v}}(s_1)\right| \\
&\le \frac{1}{n\cdot 2^w} \sum_{a\in\{0,1\}^w} \sum_{i\in T\cap[\ell]} \sum_{s_1\subseteq T:|s_1|=k} \left|\widehat{B^{(i,a)}_{\to v}}(s_1)\right| \\
&\le \frac{n\cdot 2^w\cdot \beta/2^k}{n\cdot 2^w} \qquad\qquad \text{(using Claim 6.5 and Fact 3.3)} \\
&\le \frac{\rho}{n\cdot 2^w} \qquad\qquad \text{(by our choice of } \beta\text{)}
\end{aligned}
$$

This completes the induction, and hence the claim follows. □

□

6.2 The Final Generator

We are now ready to prove our main theorem which we restate here.

Theorem (Theorem 1, restated)**.** *For all $n > 0$ and $w \leq \log n$, there exists an explicit pseudorandom generator for the class of unordered, read-once, oblivious branching programs of length n and width w with seed length $O((\log^{w+1} n)\log\log n)$ and error $1/n^{O(1)}$.*

Theorem 6.2 combined with a hybrid argument implies that the following distribution ε-fools any unordered ROBP of length n and width w:

Choose p, δ, k and σ so that they satisfy Theorem 6.2. Let $m = \min\{n, 2(\ln n)/p\} \leq O(\log n)^{w-1} \cdot \log(n/\varepsilon)$ be the number of iterations of the pseudorandom generator.

- Choose disjoint sets $T_1, ..., T_m \subseteq [n]$ where $T_i \subseteq [n] \setminus \cup_{j=1}^{i-1} T_j$ is selected according to a δ-almost $2k$-wise independent distribution
- Choose $X_1, ..., X_m$, where $X_i \in \{0,1\}^{T_i}$, each independently at random according to a σ-biased distribution
- Let $Y \in \{0,1\}^{[n]\setminus \cup_i T_i}$ be chosen using a σ-biased distribution.
- Output $X \in \{0,1\}^m$ such that $X|_{T_i} = X_i$ and $X|_{T \setminus \cup T_i} = Y$.

First, we show that by the choice of m, with probability at least $1 - \varepsilon/4$, the size of the remaining set of variables $[n] \setminus \bigcup_i |T_i|$ is at most $O(\log(1/\varepsilon))$. This is true because for any fixed set $I \subseteq [n]$ of variables, the probability that all x_i for $i \in I$ are not chosen for any of $T_1, \ldots, T_m$ is at most

$$((1-p)^{|I|} + \delta)^m \leq (e^{-p|I|} + \delta)^m \leq (e^{-p|I|}(1+2\delta))^m \leq (e^{-p|I|+2\delta})^m \leq e^{-p|I|m+2\delta m} \leq O(1/n^{2|I|}).$$

Thus, by union bounding over all sets I of size $\log(1/\varepsilon)$ with probability at most $\binom{n}{\log(1/\varepsilon)} \cdot n^{-2\log(1/\varepsilon)} \leq \varepsilon/4$ there exists such a set which is not covered by $T_1, \ldots, T_m$. Assuming that Y is defined over at most $\log(1/\varepsilon)$ variables, it is $\varepsilon/4$-fooled by a σ-biased distribution (by the choice of $\sigma = \mathrm{poly}(\varepsilon/n)$). Applying Theorem 6.2 for m steps in a hybrid argument, with each step incurring at most $\varepsilon/(2n) \leq \varepsilon/(2m)$ error, proves the correctness of the generator.

It remains to argue that we can achieve the above distribution with a short seed-length. Using an explicit construction of [AGHP92], a δ-almost (k,p)-wise distribution can be generated using $O(k \cdot \log(1/p) + \log(\frac{\log n}{\delta})) = O(w \log(n/\varepsilon) \log\log(n/\varepsilon))$ random bits. Naor and Naor [NN93] gave an explicit construction of a σ-biased distribution over n bits that uses $O(\log(n/\sigma)) = O(\log(n/\varepsilon))$ random bits.

Putting things together, our generator can be explicitly constructed with seed-length

$$O(\log n)^{w-1} \cdot \log^2(n/\varepsilon) \cdot \log\log(n/\varepsilon)$$

and seed-length $(O(\log n)^{w+1} \cdot \log\log n)$ for $\varepsilon = 1/\mathrm{poly}(n)$.

7 Fourier Bounds for General Branching Programs

Lemma 7.1. *Let B be a length-n, width-w ROBP with $w \leq n$. Then*

$$\sum_{s:|s|=k} \left|\widehat{B}(s)\right| \leq O(\log n)^{(w-2)k} \cdot \log^k \left(4 \cdot (nw)^k / \mathbf{Pr}[B(x)=1]\right) \cdot \mathbf{Pr}[B(x)=1].$$

for all $k \in [n]$.

Proof. Let $t = 2 \cdot (C \cdot \log n)^{w-2}$ where C is the constant from Lemma 4.3. We prove, by induction on $k \in \mathbb{N}$, that for all ROBP B of length at most n width at most w, it holds that

$$\sum_{s:|s|=k} \left|\widehat{B}(s)\right| \le t^k \cdot \log^k \left(4 \cdot (nw)^k / \mathbf{Pr}[B(x)=1]\right) \cdot \mathbf{Pr}[B(x)=1].$$

The base case is given by Lemma 4.3. Assuming the claim holds for k, we show that it holds for $k+1$. Recall the definitions of $B_{\to v}$, $B_{v\to}$ and p_v. Using Equation (1) we have

$$\begin{aligned}
\sum_{s:|s|=k+1} \left|\widehat{B}(s)\right| &\le \sum_{i\in[n]} \sum_{\substack{|s'|=k,\\ v\in V_i}} \left|\widehat{B_{\to v}}(s')\right| \cdot \left|\widehat{B_{v\to}}(i)\right| \\
&\le \sum_{i\in[n]} \sum_{v\in V_i} |\widehat{B_{v\to}}(i)| \cdot \sum_{|s'|=k} \left|\widehat{B_{\to v}}(s')\right| \\
&\le \sum_{i\in[n]} \sum_{v\in V_i} |\widehat{B_{v\to}}(i)| \cdot \left(t^k \cdot \log(4(nw)^k / \mathbf{Pr}[B_{\to v}(x)=1]) \cdot \mathbf{Pr}[B_{\to v}(x)=1]\right) \\
&\qquad\qquad \text{(induction on } B_{\to v} \text{ and sets of size } k) \\
&\le t^k \cdot \sum_{i\in[n]} \sum_{v\in V_i} |\widehat{B_{v\to}}(i)| \cdot \log(4(nw)^k/p_v)^k \cdot p_v \ . \quad (\text{since } p_v = \mathbf{Pr}[B_{\to v}(x)=1])
\end{aligned}$$

We break the sum of the right hand side according to whether or not $p_v \le p/(nw)$. The partial sum over vertices with $p_v > p/nw$ is at most

$$\begin{aligned}
&\sum_{i\in[n], v\in V_i, p_v > p/nw} |\widehat{B_{v\to}}(i)| \cdot \log(4(nw)^k/p_v)^k \cdot p_v \\
&\qquad \le \sum_{i\in[n], v\in V_i, p_v > p/nw} |\widehat{B_{v\to}}(i)| \cdot \log(4(nw)^{k+1}/p)^k \cdot p_v \\
&\qquad = \log(4(nw)^{k+1}/p)^k \cdot \sum_{i\in[n], v\in V_i, p_v > p/nw} |\widehat{B_{v\to}}(i)| \cdot p_v \\
&\qquad \le \log(4(nw)^{k+1}/p)^k \cdot p \cdot (C \cdot \log n)^{w-2} \cdot \log(1/p) \qquad \text{(Corollary 4.4)} \\
&\qquad \le \log(4(nw)^{k+1}/p)^{k+1} \cdot p \cdot (C \cdot \log n)^{w-2} \ .
\end{aligned}$$

The partial sum over vertices with $p_v \le p/(nw)$ is at most

$$\begin{aligned}
&\sum_{i\in[n], v\in V_i, p_v \le p/nw} |\widehat{B_{v\to}}(i)| \cdot \log(4(nw)^k/p_v)^k \cdot p_v \\
&\qquad \le \sum_{i\in[n], v\in V_i, p_v \le p/nw} |\widehat{B_{v\to}}(i)| \cdot \log(4(nw)^{k+1}/p)^k \cdot \frac{p}{nw} \ \le \ \log(4(nw)^{k+1}/p)^k \cdot p,
\end{aligned}$$

where the first inequality follows since $\log(4(nw)^k/x)^k \cdot x$ is monotone increasing for $x \in [0,1]$, and the second inequality follows since $|\widehat{B_{v\to}}(i)| \le 1$ and there are at most nw terms in the sum. Thus the total sum is at most

$$\sum_{s:|s|=k+1} \left|\widehat{B}(s)\right| \le t^k \cdot \log(4(nw)^{k+1}/p)^{k+1} \cdot p \cdot ((C \cdot \log n)^{w-2} + 1)$$

$$\leq t^k \cdot \log(4(nw)^{k+1}/p)^{k+1} \cdot p \cdot (2 \cdot (C \cdot \log n)^{w-2})$$
$$= t^{k+1} \cdot \log(4(nw)^{k+1}/p)^{k+1} \cdot p \qquad \text{(Choice of } t)$$

□

Corollary 7.2. *Let B be a length-n, width-w ROBP with $w \leq n$. Then*

$$\sum_{s:|s|=k} \left|\widehat{B}(s)\right| \leq O(\log n)^{(w-1)k} \cdot k^k,$$

for all $k \in [n]$.

Proof. Note that the bound from Lemma 7.1 is monotone increasing in p, thus

$$\sum_{s:|s|=k} \left|\widehat{B}(s)\right| \leq O(\log n)^{(w-2)k} \cdot \log^k\left(4 \cdot (nw)^k\right) = O(\log n)^{(w-2)k} \cdot O(k \log n)^k. \quad \square$$

Theorem (Theorem 2, restated)**.** *Let B be a length-n, width-w ROBP with $w \leq n$. Then*

$$\sum_{s:|s|=k} \left|\widehat{B}(s)\right| \leq O(\log n)^{wk} ,$$

for all $k \in [n]$.

This proves Conjecture 8.1 from [RSV13]. We believe that the right bound should be

$$\sum_{s:|s|=k} \left|\widehat{B}(s)\right| \leq O(\log n)^{(w-2)k}$$

for all $k \in [n]$. For $w = 3$, Steinke et al. [SVW14] proved $L_{1,k}(B) \leq O(\log n)^k \cdot n^2$. As for lower bounds, for $w = 3$, Mansour [Man95, Appendix A] and Steinke et al. [SVW14, Appendix C] analyzed the Tribes function and showed that $\forall k \in [n] : \sum_{|s|=k} |\widehat{B}(s)| \geq \Omega(\log n/\log k)^k$.

Proof. For $k \leq 2 \cdot \log(nw)$ the theorem follows from the bound in Corollary 7.2. Furthermore, for $k \in [\log(nw), 2\log(nw)]$ we have

$$\sum_{s:|s|=k} \left|\widehat{B}(s)\right| \leq O(\log n)^{wk} \leq O(\log n)^{wk} \cdot \frac{2^k}{nw} .$$

We prove by induction on $k \geq \log(nw)$ that $\sum_{s:|s|=k} \left|\widehat{B}(s)\right| \leq O(\log n)^{wk} \cdot \frac{2^k}{nw}$. For $k \geq 2 \cdot \log(nw)$, denote by $k' = k - \log(nw)$ and $k'' = \log(nw)$. As done in [RSV13, SVW14], we split the sum over sets of size k to the sum over sets of size k' in the prefix of B times the sum over sets of size k'' in the suffix of B, for all n possible partitions of B into two parts.

$$\begin{aligned}
\sum_{|s|=k} |\widehat{B}(s)| &\leq \sum_{i=1}^{n} \sum_{v \in V_i} \sum_{\substack{s' \subseteq \{1,\ldots,i-1\},\\ |s'|=k'}} \sum_{\substack{s'' \subseteq \{i,\ldots,n\},\\ |s''|=k''}} |\widehat{B_{\to v}}(s') \cdot \widehat{B_{v\to}}(s'')| \\
&\leq \sum_{i=1}^{n} \sum_{v \in V_i} O(\log n)^{wk'} \cdot \frac{2^{k'}}{nw} \cdot O(\log n)^{wk''} \cdot \frac{2^{k''}}{nw} \\
&= 2^k \cdot O(\log n)^{wk} \cdot \sum_i \sum_{v \in V_i} \frac{1}{nw} \cdot \frac{1}{nw} = O(\log n)^{wk} \cdot \frac{2^k}{nw} .
\end{aligned}$$

□

Acknowledgements

We would like to thank Michael P. Kim, Shachar Lovett, Raghu Meka, Ran Raz, Salil Vadhan and David Zuckerman for very helpful conversations.

References

[AGHP92] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple construction of almost k-wise independent random variables. *Random Structures and Algorithms*, 3(3):289–304, 1992.

[AW85] M. Ajtai and A. Wigderson. Deterministic simulation of probabilistic constant depth circuits. In *FOCS*, pages 11–19, 1985.

[BDVY13] A. Bogdanov, Z. Dvir, E. Verbin, and A. Yehudayoff. Pseudorandomness for width-2 branching programs. *Theory of Computing*, 9:283–293, 2013.

[BPW11] A. Bogdanov, P. A. Papakonstantinou, and A. Wan. Pseudorandomness for read-once formulas. In R. Ostrovsky, editor, *FOCS*, pages 240–246. IEEE, 2011.

[BRRY10] M. Braverman, A. Rao, R. Raz, and A. Yehudayoff. Pseudorandom generators for regular branching programs. In *Proceedings of the 51st annual FOCS*, pages 40–47, 2010.

[BV10] J. Brody and E. Verbin. The coin problem and pseudorandomness for branching programs. In *Proceedings of the 51st annual FOCS*, pages 30–39, 2010.

[CRSW13] L. E. Celis, O. Reingold, G. Segev, and U. Wieder. Balls and bins: Smaller hash families and faster evaluation. *SIAM Journal on Computing*, 42(3):1030–1050, 2013.

[De11] A. De. Pseudorandomness for permutation and regular branching programs. In *IEEE Conference on Computational Complexity*, pages 221–231, 2011.

[EIO02] L. Engebretsen, P. Indyk, and R. O'Donnell. Derandomized dimensionality reduction with applications. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 705–712, 2002.

[GMR$^+$12] P. Gopalan, R. Meka, O. Reingold, L. Trevisan, and S. P. Vadhan. Better pseudorandom generators from milder pseudorandom restrictions. In *FOCS*, pages 120–129, 2012.

[HHR11] I. Haitner, D. Harnik, and O. Reingold. On the power of the randomized iterate. *SIAM J. Comput.*, 40(6):1486–1528, December 2011.

[HVV06] A. Healy, S. Vadhan, and E. Viola. Using nondeterminism to amplify hardness. *SIAM Journal on Computing*, 35(4):903–931, 2006.

[IMZ12] R. Impagliazzo, R. Meka, and D. Zuckerman. Pseudorandomness from shrinkage. In *Proceedings of the 53rd annual FOCS*, pages 111–119, 2012.

[Ind06] P. Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, 2006.

[INW94] R. Impagliazzo, N. Nisan, and A. Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the 26th annual STOC*, pages 356–364, 1994.

[KNP11] M. Koucký, P. Nimbhorkar, and P. Pudlák. Pseudorandom generators for group products: extended abstract. In *STOC*, pages 263–272, 2011.

[Man95] Y. Mansour. An $O(n^{\log \log n})$ learning algorithm for DNF under the uniform distribution. *J. Comput. Syst. Sci.*, 50(3):543–550, 1995.

[Nis92] N. Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

[NN93] J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. on Computing*, 22(4):838–856, 1993.

[NZ96] N. Nisan and D. Zuckerman. Randomness is linear in space. *J. of Computer and System Sciences*, 52(1):43–52, 1996.

[O'D14] R. O'Donnell. *Analysis of boolean functions.* Cambridge University Press, 2014.

[RSV13] O. Reingold, T. Steinke, and S. Vadhan. Pseudorandomness for regular branching programs via Fourier analysis. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 655–670. Springer, 2013.

[Siv02] D. Sivakumar. Algorithmic derandomization via complexity theory. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 619–626, 2002.

[Ste13] J. P. Steinberger. The distinguishability of product distributions by read-once branching programs. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013*, pages 248–254, 2013.

[SVW14] T. Steinke, S. Vadhan, and A. Wan. Pseudorandomness and fourier growth bounds for width-3 branching programs. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, page 885, 2014.

[SZ95] M. Saks and D. Zuckerman. Personal Communication, 1995.

[SZ99] M. E. Saks and S. Zhou. $\mathrm{BP_H}$ Space(S) $\subseteq$ DSPACE($\mathrm{S}^{3/2}$). *J. Comput. Syst. Sci.*, 58(2):376–403, 1999.

[Tzu09] Y. Tzur. Notions of weak pseudorandomness and GF(2^n)-polynomials. Master's thesis, Master's thesis, Weizmann Institute of Science, Rehovot, Israel, 2009.