Pseudorandom Generators for Unbounded-Width Permutation Branching Programs

William M. Hoza
University of Texas at Austin, TX, USA
https://williamhoza.com/tcs/
whoza@utexas.edu

Edward Pyne
Harvard University, Cambridge, MA, USA
https://sites.google.com/view/tedpyne/
epyne@college.harvard.edu

Salil Vadhan
Harvard University, Cambridge, MA, USA
https://salil.seas.harvard.edu/
salil_vadhan@harvard.edu

Abstract

We prove that the Impagliazzo-Nisan-Wigderson [9] pseudorandom generator (PRG) fools ordered (read-once) permutation branching programs of *unbounded width* with a seed length of $\widetilde{O}(\log d + \log n \cdot \log(1/\varepsilon))$, assuming the program has only one accepting vertex in the final layer. Here, n is the length of the program, d is the degree (equivalently, the alphabet size), and ε is the error of the PRG. In contrast, we show that a randomly chosen generator requires seed length $\Omega(n \log d)$ to fool such unbounded-width programs. Thus, this is an unusual case where an explicit construction is "better than random."

Except when the program's width w is very small, this is an improvement over prior work. For example, when $w = \text{poly}(n)$ and $d = 2$, the best prior PRG for permutation branching programs was simply Nisan's PRG [15], which fools general ordered branching programs with seed length $O(\log(wn/\varepsilon) \log n)$. We prove a seed length lower bound of $\widetilde{\Omega}(\log d + \log n \cdot \log(1/\varepsilon))$ for fooling these unbounded-width programs, showing that our seed length is near-optimal. In fact, when $\varepsilon \leq 1/\log n$, our seed length is within a constant factor of optimal. Our analysis of the INW generator uses the connection between the PRG and the derandomized square of Rozenman and Vadhan [20] and the recent analysis of the latter in terms of unit-circle approximation by Ahmadinejad et al. [1].

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization

Keywords and phrases Pseudorandom generators, permutation branching programs

Digital Object Identifier 10.4230/LIPIcs.ITCS.2021.7

Funding *William M. Hoza*: Supported by the NSF GRFP under grant DGE-1610403 and a Harrington Fellowship from UT Austin.
Edward Pyne: Supported by NSF grant CCF-1763299.
Salil Vadhan: Supported by NSF grant CCF-1763299 and a Simons Investigator Award.

Acknowledgements We thank Jack Murtagh for collaboration at the start of this research. The first author thanks Dean Doron for insightful and relevant discussions about the works by Murtagh et al. [13, 14]. We thank Shyam Narayanan, Dean Doron and David Zuckerman for valuable comments on a draft of this paper.

12th Innovations in Theoretical Computer Science Conference (ITCS 2021).
Editor: James R. Lee; Article No. 7; pp. 7:1–7:20
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Randomness, like time or space, is a computational resource. All else being equal, it is best to use as few random bits as possible. A *pseudorandom generator* (PRG) is a tool for reducing the number of random bits used by some computational process.

▶ **Definition 1.1.** *Let $\mathcal{F}$ be a class of functions $B\colon [d]^n \to \{0,1\}$. An **$\varepsilon$-PRG** for $\mathcal{F}$ is a function $G\colon \{0,1\}^s \to [d]^n$ such that for every $B \in \mathcal{F}$,*

$$|\Pr[B(U_{[d]^n}) = 1] - \Pr[B(G(U_{\{0,1\}^s})) = 1]| \le \varepsilon,$$

*where U_S is the uniform distribution over the set S. The value s is the **seed length** of the PRG.*

Motivated by the goal of derandomizing small-space computation, a long line of research has studied PRGs for classes $\mathcal{F}$ of functions computable by *branching programs.*

▶ **Definition 1.2.** *An ordered **branching program** B of length n, width w and degree d computes a function $B : [d]^n \to \{0,1\}$. At time step $t \in [n]$, the program maintains a state in $[w]$, reads the next symbol σ_t of the input $\sigma \in [d]^n$ and updates its state according to a transition function $W_t : [w] \times [d] \to [w]$. We allow the transition function W_t to be different at each time step.*

Moreover, there is an initial state $v_{start} \in [w]$ and a single accept state $v_{end} \in [w]$. Let u be the final state of the branching program on input σ. If $u = v_{end}$ the branching program accepts, denoted $B(\sigma) = 1$. For any other final state the program rejects, denoted $B(\sigma) = 0$.

We can represent a branching program as a graph, with $n+1$ layers and w vertices per layer corresponding to the states of the program at each step. For all $t \in [n]$, for state s in layer $t-1$ and s' in layer t, we add edge (s, s') with label $\sigma_t \in [w]$ if $W_t(s, \sigma_t) = s'$. An ordered read-once branching program of length n and width w can compute the output of an algorithm that uses $\log w$ bits of memory and n random bits, by taking the state at each layer as the contents of memory at that time. Unusually, we will consider branching programs where the width is unbounded (e.g., it can even be $w = d^n$), albeit with the restriction of being a *permutation branching program.*

▶ **Definition 1.3.** *A **permutation branching program** is an ordered branching program where for all $t \in [n]$ and $\sigma \in [d]$, $W_t(\cdot, \sigma)$ is a permutation. This can be thought of as the computation being time-reversible.*

Note that with this restriction the graph representation consists of $n+1$ layers where each layer is the union of d perfect matchings, with each matching corresponding to a distinct input symbol.

Restricted classes of branching programs, including permutation branching programs [22, 5, 10], have received attention largely because of the lack of progress on designing PRGs for general length-n width-n branching programs since the work of Nisan three decades ago [15]. There has also been work on permutation branching programs where the input is read in an arbitrary order [17, 3]. Our main theorem is that there is an explicit PRG fooling unbounded-width permutation branching programs with seed length that is nearly logarithmic in n *and has no dependence on the width w*:

▶ **Theorem 1.4** (Main Theorem). *For all $n, d \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicitly computable ε-PRG $G\colon \{0,1\}^s \to [d]^n$ for permutation branching programs of length n, degree d, and arbitrary width. This PRG has seed length*

$$O\left(\log d + \log n \cdot (\log\log n + \log(1/\varepsilon)\right).$$

In contrast, we show that a randomly chosen generator requires seed length $\Omega(n \log d)$ to fool such unbounded-width programs. Thus, this is an unusual case where an explicit construction is "better than random." (See Section 1.3 for more discussion.)

The PRG is an instantiation of the Impagliazzo-Nisan-Wigderson (INW) generator [9]. The proof uses the interpretation of the INW generator in terms of the derandomized square for consistently labeled graphs, introduced by Rozenman and Vadhan [20], and the analysis of the derandomized square in terms of unit-circle approximation by Ahmadinejad et al. [1].

We emphasize that our definition of permutation branching program only allows one accepting vertex. This assumption is crucial: a permutation branching program with unbounded width and an unbounded number of accepting vertices can compute any Boolean function on $[d]^n$, so nontrivial PRGs for that model do not exist. That being said, a program with a accepting vertices can be written as a sum of a programs with one accepting vertex each, so our PRG fools such a permutation branching program with seed length

$$O(\log d + \log n \cdot (\log \log n + \log(a/\varepsilon))).$$

1.1 Prior Work on the Derandomized Square

In the paper introducing the derandomized square [20], Rozenman and Vadhan showed how to use it to decide undirected connectivity in deterministic log space, giving another proof of Reingold's famous theorem [16]. As another application, they showed how to take a (polynomially long) pseudorandom walk through a regular, aperiodic directed graph in such a way that the final vertex is distributed nearly uniformly (i.e., is close to the stationary distribution of a truly random walk), matching a result of Reingold, Trevisan, and Vadhan [18]. As mentioned previously, they observed that this pseudorandom walk is described by the INW generator, assuming the graph is "consistently labeled" (see Definition 3.4). However, their analysis does not show how to approximate short random walks (e.g., shorter than the mixing time).

In a pair of relatively recent works [13, 14], Murtagh et al. showed how to approximate (in some respects) random walks of any length n, even if n is much smaller than the graph's mixing time. These algorithms are only for undirected graphs, but the recent work by Ahmadinejad et al. [1] handles the more general case of Eulerian digraphs (as well as getting stronger results for undirected graphs). Among other tools, all three of these papers [13, 14, 1] use the derandomized square.

Fooling branching programs amounts to approximating *bounded-length* random walks through *directed* graphs, which is why we rely on Ahmadinejad et al.'s results [1] for our theorem. One of their results is a deterministic non-black-box algorithm for estimating the acceptance probability of a given polynomial-width "regular" branching program in space $\widetilde{O}(\log n)$ to within error $\varepsilon = 1/\operatorname{poly}(n)$. Our theorem solves the more challenging black-box derandomization problem, although it only works for permutation branching programs and we have a worse dependence on the error parameter ε.

1.2 Prior PRGs for Permutation Branching Programs

Our PRG is superior to prior generators for permutation branching programs[1] when the width of the branching program is not small. Previous work has focused on the constant-width case. In that regime, the best PRG for permutation branching programs is due to Steinke [22].

[1] In this discussion of prior work, we focus on the case $d = 2$ for simplicity.

He achieves seed length $O(w^4 \log w \log n + \log n \log(1/\varepsilon))$, which is better than our seed length by a factor of $\log \log n$. For larger widths up to $w = \mathrm{poly}(n)$, the best prior PRG for permutation branching programs is by Braverman et al. [2], who gave a PRG for regular branching programs with seed length

$$O(\log w \log n + \log n \cdot (\log \log n + \log(1/\varepsilon))).$$

Note that when $w = \mathrm{poly}(n)$ and $\varepsilon = \Omega(1)$, Braverman et al.'s PRG has seed length $\Theta(\log^2 n)$, just like Nisan's PRG [15], whereas our PRG has seed length $\widetilde{O}(\log n)$. The case $w = \mathrm{poly}(n)$ is arguably the most important case, because polynomial-width ordered branching programs correspond to uniform randomized algorithms that always halt. Recall that low-error PRGs for polynomial-width regular branching programs suffice for derandomizing all of **RL** [18].

When the width is even larger than $\mathrm{poly}(n)$, the best prior PRG is by De [5]. De's work is focused on the constant-width case, but he also gave a generator with seed length $O(\log(n/\varepsilon) \log n)$ independent of w.

1.3 Failure of the Probabilistic Method

There is something counterintuitive about the superpolynomial-width regime. Recall that for typical models of computation, including polynomial-width degree-2 branching programs, it is straightforward to show that there exists a nonexplicit PRG with seed length $O(\log(n/\varepsilon))$, because a random function is a good PRG. Furthermore, it is typically fairly trivial to prove a matching $\Omega(\log(n/\varepsilon))$ lower bound. The main challenge, in most cases, is to devise an explicit construction matching the parameters of the probabilistic existence proof.

However, the standard nonexplicit existence argument is not applicable to unbounded-width permutation branching programs, because they can compute doubly-exponentially many distinct functions; in particular, we show (Lemma 5.1) that they can compute every Boolean function $B(x, y)$ that tests whether $\pi(x) = y$ for a permutation $\pi\colon [d]^{n/2} \to [d]^{n/2}$. And indeed, as mentioned previously, for seed length less than $(n \log d)/4$, we show (Theorem 5.2) that a random function is *not* a good PRG for this model. The reason is that when a generator is chosen at random, with high probability, there is some permutation π such that every output (x, y) of the generator satisfies $\pi(x) = y$.

Since the probabilistic method fails here, it might be surprising that there even *exists* a PRG with near-logarithmic seed length, let alone our explicit construction. Intuitively, the INW generator manages to outperform the probabilistic method because the second half of the INW generator's output is *information-theoretically* unpredictable given the first half, and vice versa.

We remark that another family of unbounded-width ordered branching programs has been studied previously: "monotone" branching programs. From Meka and Zuckerman's work [12], it follows that a random function *is* a good PRG for unbounded-width monotone branching programs. In that respect, the model we study is more unusual.

1.4 The Optimal Seed Length

These considerations raise the question of what the optimal seed length is for our model. We prove (Theorem 6.1) that any PRG for unbounded-width permutation branching programs must have seed length at least $\Omega(\log d + \log n \cdot \log(1/\varepsilon))$, provided ε is not extremely small.[2] Thus, our explicit PRG's seed length is near-optimal. In fact, although the lower bound gets

[2] E.g., any $\varepsilon \geq \exp(-(n \log d)^{0.99})$ is large enough.

slightly weaker when ε is extremely small, we give a matching refinement of our *upper* bound in that regime (see Corollary 4.9), providing an explicit PRG with asymptotically optimal seed length whenever $\varepsilon \leq 1/\log n$.

To the best of our knowledge, this is the first known case where, e.g., some seed length s is sufficient for a constant-error PRG, but seed length $O(s + \log n)$ is not sufficient to achieve error $1/n$. In the context of fooling shallow circuits, similar lower bounds were proven previously for restricted classes of PRGs such as k-wise independent distributions [11] or small-bias distributions [6], but our lower bound holds for any PRG whatsoever. Our lower bound uses basic tools from matching theory and information theory.

On the other hand, we show (Theorem 7.1) that a random function is at least a good *hitting set* generator (HSG); the optimal seed length for nonexplicit HSGs for unbounded-width permutation branching programs is $\Theta(\log(nd/\varepsilon))$. This is the first case we are aware of where there is a large gap between the best possible PRGs and the best possible HSGs.

1.5 Organization

In Section 2, we introduce measures of spectral approximation for matrices and basic linear algebra facts. In Section 3, we introduce the derandomized square, and recall two theorems relating the square to unit-circle approximation, then prove repeated derandomized squaring provides a suitable quality approximation. In Section 4, we use the bounds on repeated derandomized squares to analyze the INW generator. In Section 5, we prove that a random function with seed length less than $(n \log d)/4$ does not fool unbounded-width permutation branching programs. In Section 6, we prove our lower bound on the seed length of any PRG for these programs. Finally, in Section 7, we identify the optimal seed length for nonexplicit HSGs for these programs.

2 Spectral Approximation Preliminaries

We first introduce basic notation and recall two measures of closeness of approximation for matrices, complex spectral approximation and unit-circle approximation.

- For a complex number $z \in \mathbb{C}$ we write z^* to denote the complex conjugate of z and $|z|$ to denote the magnitude of z.
- For a matrix $A \in \mathbb{C}^{N \times N}$ we write A^* to denote its conjugate transpose and write $U_A = (A + A^*)/2$ to denote its **symmetrization**.
- We say a Hermitian matrix A is **positive semidefinite** (PSD) or write $A \succeq 0$ if $x^*Ax \geq 0$ for all $x \in \mathbb{C}^N$. For two Hermitian matrices A, B, we use $A \succeq B$ to denote $A - B \succeq 0$ and define $\preceq$ analogously.

▶ **Definition 2.1** (Complex Spectral Approximation [1]). *For $A, B \in \mathbb{C}^{N \times N}$ and $\varepsilon > 0$, we say A is a* ***complex ε-approximation*** *of B, denoted $A \approx_\varepsilon B$, if*

$$\forall x, y \in \mathbb{C}^N, \quad |x^*(B - A)y| \leq \frac{\varepsilon}{2}(||x||^2 + ||y||^2 - x^*U_Bx - y^*U_By).$$

For two N-vertex digraphs $\widetilde{\mathbf{G}}, \mathbf{G}$ with random walk matrices A, B, write $\widetilde{\mathbf{G}} \approx_\varepsilon \mathbf{G}$ if $A \approx_\varepsilon B$.

We now recall the stronger notion that we will use for analyzing the generator.

▶ **Definition 2.2** (Unit-Circle Approximation [1]). *For $A, B \in \mathbb{C}^{N \times N}$ and $\varepsilon > 0$, we say A is a* ***unit-circle ε-approximation*** *of B, denoted $A \overset{\circ}{\approx}_\varepsilon B$, if*

$$\forall x, y \in \mathbb{C}^N, \quad |x^*(B - A)y| \leq \frac{\varepsilon}{2}(||x||^2 + ||y||^2 - |x^*Bx + y^*By|).$$

For two N-vertex digraphs $\widetilde{\mathbf{G}}, \mathbf{G}$ with random walk matrices A, B, write $\widetilde{\mathbf{G}} \overset{\circ}{\approx}_\varepsilon \mathbf{G}$ if $A \overset{\circ}{\approx}_\varepsilon B$.

Including the magnitude operation in the right hand side forces the approximation to be exact for all eigenspaces with eigenvalues of complex magnitude 1, and this property is essential for the preservation of approximation under high powers. The unit-circle approximation is developed in [1]. We rely on a convenient equivalence between unit-circle approximation and complex approximation:

▶ **Lemma 2.3** ([1] Lemma 3.8). *Let $A, B \in \mathbb{C}^{N \times N}$ and $\varepsilon > 0$. Then $A \overset{\circ}{\approx}_\varepsilon B$ if and only if for all $z \in \mathbb{C}$ with $|z| = 1$, $zA \approx_\varepsilon zB$.*

We will also use this basic result about complex approximation. Note that Cohen et al. [4] prove the analogous statement where complex numbers are replaced with reals.

▶ **Lemma 2.4.** *Let $A, B \in \mathbb{C}^{N \times N}$ where $A \approx_\varepsilon B$. Then $(1-\varepsilon)U_{I-B} \preceq U_{I-A} \preceq (1+\varepsilon)U_{I-B}$.*

Proof. Let arbitrary $x \in \mathbb{C}^N$. Bounding the gap between the symmetrizations via the definition of complex approximation gives

$$\begin{aligned}|x^*U_{I-B}x - x^*U_{I-A}x| &= \left|\frac{1}{2}(x^*(B+B^*)x - x^*(A+A^*)x)\right| \\ &\leq \left|\frac{1}{2}x^*(B-A)x\right| + \left|\frac{1}{2}x^*(B^*-A^*)x\right| \\ &\leq \frac{1}{2}\varepsilon(||x||^2 - x^*U_Bx) + \frac{1}{2}\varepsilon(||x||^2 - x^*U_Bx) \\ &= \varepsilon \cdot x^*U_{I-B}x.\end{aligned}$$

This directly implies $x^*U_{I-A}x - (1-\varepsilon)x^*U_{I-B}x \geq 0$ and $(1+\varepsilon)x^*U_{I-B}x - x^*U_{I-A}x \geq 0$. Since x was arbitrary we are done. ◀

We now state an approximate triangle inequality for unit-circle approximation, which will be a tool for bounding the error of the generator. Previously, Cohen et al. [4] proved a similar lemma regarding the real analogue of complex approximation.

▶ **Lemma 2.5** (Quasi-Triangle Inequality). *If $C \overset{\circ}{\approx}_{\varepsilon_2} B \overset{\circ}{\approx}_{\varepsilon_1} A$ then $C \overset{\circ}{\approx}_{\varepsilon_1+\varepsilon_2+\varepsilon_1\varepsilon_2} A$.*

Proof. Let $z \in \mathbb{C}$ satisfy $|z| = 1$, and let $x, y \in \mathbb{C}^N$ be arbitrary. Since $B \overset{\circ}{\approx}_{\varepsilon_1} A$, by Lemma 2.3, $zB \approx_{\varepsilon_1} zA$, so

$$|x^*(A-B)y| \leq \frac{\varepsilon_1}{2}(x^*U_{I-zA}x + y^*U_{I-zA}y).$$

Similarly, since $C \overset{\circ}{\approx}_{\varepsilon_2} B$,

$$\begin{aligned}|x^*(B-C)y| &\leq \frac{\varepsilon_2}{2}(x^*U_{I-zB}x + y^*U_{I-zB}y) \\ &\leq \frac{\varepsilon_2}{2} \cdot (1+\varepsilon_1) \cdot (x^*U_{I-zA}x + y^*U_{I-zA}y)\end{aligned}$$

where the second inequality follows from Lemma 2.4. Therefore,

$$\begin{aligned}|x^*(A-C)y| &\leq |x^*(A-B)y| + |x^*(B-C)y| \\ &\leq \left(\frac{\varepsilon_1}{2} + \frac{\varepsilon_2}{2} + \frac{\varepsilon_1\varepsilon_2}{2}\right) \cdot (x^*U_{I-zA}x + y^*U_{I-zA}y).\end{aligned}$$

Since x and y were arbitrary, this shows that $zC \approx_{\varepsilon_1+\varepsilon_2+\varepsilon_1\varepsilon_2} zA$. Since z was arbitrary, we are done by Lemma 2.3. ◀

▶ **Corollary 2.6** (Iterated Quasi-Triangle Inequality). *Suppose $\delta \leq 1$ and*

$$A_0 \overset{\circ}{\approx}_\delta A_1 \overset{\circ}{\approx}_\delta \ldots \overset{\circ}{\approx}_\delta A_\ell.$$

Then $A_0 \overset{\circ}{\approx}_\varepsilon A_\ell$ with $\varepsilon = \ell\delta/(1-\delta)^2$.

Proof. Applying Lemma 2.5 inductively, we get a bound of

$$\sum_{i=0}^{\ell}(\ell - i) \cdot \delta^{i+1} = \frac{\ell\delta - \delta^2 \cdot (\ell + 1 - \delta^\ell)}{(1-\delta)^2} \leq \frac{\ell\delta}{(1-\delta)^2}.$$ ◀

Finally, we give a basic result used to relate unit-circle to entrywise approximation for the final generator analysis.

▶ **Proposition 2.7.** *Given $A, B \in \mathbb{C}^{N\times N}$ so that $A \overset{\circ}{\approx}_\varepsilon B$, for all indices $u, v \in [N]$, $|A_{u,v} - B_{u,v}| \leq \varepsilon$.*

Proof. Let e_u, e_v be the standard basis vectors with ones in coordinates u, v respectively and apply Definition 2.2:

$$|A_{u,v} - B_{u,v}| = |e_u^*(A-B)e_v| \leq \frac{\varepsilon}{2}(||e_u||^2 + ||e_v||^2 - |e_u^* B e_u + e_v^* B e_v|) \leq \varepsilon.$$ ◀

3 Repeated Derandomized Squaring

3.1 Graph Labelings

Branching programs are closely related to graphs with *one-way labelings*.

▶ **Definition 3.1** (One-Way Labeling [20]). *A **one-way labeling** of a d-regular directed multigraph $\mathbf{G}$ assigns a label in $[d]$ to each edge (u, v) such that for every vertex u, the labels of the outgoing edges of u are distinct. If $\mathbf{G}$ has a one-way labeling, let $\mathbf{G}[u, i]$ denote the vertex v such that (u, v) is labeled i.*

One-way labelings are compatible with the operation of *powering* a graph. One step on $\mathbf{G}^n$ corresponds to n steps in $\mathbf{G}$. The formal definition follows.

▶ **Definition 3.2** (Graph Powering). *Let $\mathbf{G}$ be a d-regular directed multigraph with a one-way labeling. For $n \geq 1$, we recursively define $\mathbf{G}^n$ to be a (d^n)-regular directed multigraph on the same vertex set with a one-way labeling given by*

$$\mathbf{G}^1 = \mathbf{G}$$
$$\mathbf{G}^{n+1}[v, (e_1, e_2)] = \mathbf{G}^n[\mathbf{G}[v, e_1], e_2],$$

identifying $[d^{n+1}] = [d] \times [d^n]$.

Derandomized squaring is a way of "approximating" the powers of a graph. The derandomized squaring operation is defined in terms of graphs with additional structure, namely, a *two-way labeling*.

▶ **Definition 3.3** (Two-Way Labeling [20]). *A **two-way labeling** of a d-regular directed multigraph $\mathbf{G}$ assigns* two *labels in $[d]$ to each edge (u, v): one as an edge incident to u (the "outgoing label") and one as an edge incoming to v (the "incoming label"). We require that for every vertex v, the outgoing labels of the outgoing edges of v are distinct, and the incoming labels of the incoming edges of v are distinct. If $\mathbf{G}$ is an N-vertex graph with a two-way labeling, we define the* rotation map *[19, 20] $\mathsf{Rot}_{\mathbf{G}} \colon [N] \times [d] \to [N] \times [d]$ by letting $\mathsf{Rot}_{\mathbf{G}}(u, i) = (v, j)$ if there is an edge (u, v) with outgoing label i and incoming label j.*

Naturally, if $\mathbf{G}$ has a two-way labeling, we think of $\mathbf{G}$ as also having a one-way labeling given by the outgoing labels: $\mathsf{Rot}_{\mathbf{G}}(u,i) = (v,j) \implies \mathbf{G}[u,i] = v$. Conversely, there is a natural way to extend any *consistent* one-way labeling (defined next) to a two-way labeling.

▶ **Definition 3.4** (Consistent One-Way Labeling [8]). *A* ***consistent one-way labeling*** *of a graph $\mathbf{G}$ is a one-way labeling such that for every vertex v, the labels of the incoming edges of v are distinct. Equivalently, $\mathbf{G}[u,i] = \mathbf{G}[v,i] \implies u = v$. If $\mathbf{G}$ has a consistent one-way labeling, then we can extend $\mathbf{G}$ to a graph $\overline{\mathbf{G}}$ that has a two-way labeling given by*

$$\mathsf{Rot}_{\overline{\mathbf{G}}}(u,i) = (\mathbf{G}[u,i], i).$$

3.2 Derandomized Squaring

Now we are ready to define the derandomized square operation, introduced by Rozenman and Vadhan [20]. Let $\mathbf{G} = (V, E)$ be a regular directed multigraph. In the true square $\mathbf{G}^2$, for each vertex $v \in V$, there is a complete bipartite graph from in-neighbors of v to outneighbors of v, equivalent to all two-step walks through v. A derandomized square picks out a pseudorandom subset of such walks by correlating the two steps via edges on an expander graph $\mathbf{H}$.

▶ **Definition 3.5** (Derandomized Square [20]). *Let $\mathbf{G}$ be a directed d-regular multigraph on N vertices with a two-way labeling. Let $\mathbf{H}$ be a directed c-regular multigraph on d vertices with a one-way labeling. We define the* ***derandomized square*** *$\mathbf{G} \circledS \mathbf{H}$ to be a (cd)-regular directed multigraph on N vertices with a one-way labeling given by*

$$(\mathbf{G} \circledS \mathbf{H})[v,(i,j)] = \mathbf{G}[v', \mathbf{H}[i', j]],$$

where $(v', i') = \mathsf{Rot}_{\mathbf{G}}(v,i)$.

Note that Definition 3.5 requires $\mathbf{G}$ to have a two-way labeling, but the derandomized square $\mathbf{G} \circledS \mathbf{H}$ itself only has a one-way labeling. If we wish to apply the derandomized squaring operation a second time to approximate $\mathbf{G}^4$, we must first assign incoming labels to the edges in $\mathbf{G} \circledS \mathbf{H}$. When they introduced the derandomized square operation, Rozenman and Vadhan studied two distinct approaches for assigning incoming edge labels [20]. The first approach is to assume that we start with a graph $\mathbf{G}$ with a consistent one-way labeling. In this case, $\overline{\mathbf{G}} \circledS \mathbf{H}$ has a consistent one-way labeling as well (see Lemma 4.2). This approach is closely connected to the INW generator [9], as we will discuss in Section 4. The second approach is to assume that $\mathbf{H}$ has a two-way labeling. In this case, one can assign incoming edge labels to $\mathbf{G} \circledS \mathbf{H}$ by setting $\mathsf{Rot}_{\mathbf{G} \circledS \mathbf{H}}(v_0, (i_0, j_0)) = (v_2, (i_3, j_1))$, where

$$(v_1, i_1) = \mathsf{Rot}_{\mathbf{G}}(v_0, i_0), \qquad (i_2, j_1) = \mathsf{Rot}_{\mathbf{H}}(i_1, j_0), \qquad (v_2, i_3) = \mathsf{Rot}_{\mathbf{G}}(v_1, i_2).$$

This is the approach taken in, e.g., the recent work of Ahmadinejad et al. [1]. Note that if $\mathbf{G}$ has a consistent one-way labeling and $\mathbf{H}$ has a two-way labeling, the two approaches for assigning incoming edge labels to $\overline{\mathbf{G}} \circledS \mathbf{H}$ do not coincide.

Like previous work, we will use auxiliary graphs $\mathbf{H}$ that are good *spectral expanders*, meaning that $\lambda(\mathbf{H})$ (defined next) is small. For the purposes of this paper, an *undirected* graph is a symmetric directed graph, i.e., a directed graph such that for every edge (u,v), the reverse edge (v,u) is also present.

▶ **Definition 3.6.** *Let $\mathbf{H}$ be an undirected regular multigraph on N vertices with random walk matrix M. We define $\lambda(\mathbf{H}) = \max_{x \in \mathbb{R}^N : \langle \mathbf{1}, x \rangle = 0} ||Mx||_2 / ||x||_2$, where $\mathbf{1}$ is the all-ones vector. This is equal to the second largest eigenvalue in absolute value of M.*

Ahmadinejad et al. showed that the derandomized square is a unit-circle approximation of the true square [1]. Since the conclusion of this theorem is only a statement about the random walk matrix of $\mathbf{G} \circledS \mathbf{H}$, the theorem is oblivious to any edge labels in $\mathbf{G} \circledS \mathbf{H}$.

▶ **Theorem 3.7** ([1] Theorem 5.9)**.** *Let* $\mathbf{G}$ *be a* d*-regular directed multigraph with a two-way labeling, and let* $\mathbf{H}$ *be a* c*-regular undirected multigraph on* d *vertices with a one-way labeling. If* $\lambda(\mathbf{H}) \le \varepsilon$*, then* $\mathbf{G} \circledS \mathbf{H} \overset{\circ}{\approx}_{2\varepsilon} \mathbf{G}^2$*.*

We now use the spectral approximation measures of Section 2 to bound the error introduced by repeated derandomized squares. In the theorem below, although $\mathbf{G}_i$ has a two-way labeling, when we write $\mathbf{G}_i = \mathbf{G}_{i-1} \circledS \mathbf{H}_i$, we merely mean equality of one-way labelings. Thus, our bound applies *regardless* of how the incoming edge labels of $\mathbf{G}_{i-1} \circledS \mathbf{H}_i$ are assigned, as long as they form a valid two-way labeling.

▶ **Theorem 3.8** (Repeated Derandomized Squaring)**.** *Let* $\mathbf{G}_0, \mathbf{G}_1, \dots, \mathbf{G}_\ell$ *be directed multigraphs on* N *vertices with two-way labelings, where* $\mathbf{G}_i$ *is* $(d \cdot c^i)$*-regular. Let* $\varepsilon \in (0, 0.12)$*, and let* $\mathbf{H}_1, \dots, \mathbf{H}_\ell$ *be undirected* c*-regular multigraphs with one-way labelings, where* $\mathbf{H}_i$ *is on* $d \cdot c^{i-1}$ *vertices and* $\lambda(\mathbf{H}_i) \le \varepsilon$*. Assume that for every* $i \in [\ell]$*, we have* $\mathbf{G}_i = \mathbf{G}_{i-1} \circledS \mathbf{H}_i$*. Then* $\mathbf{G}_\ell \overset{\circ}{\approx}_{8\ell\varepsilon} \mathbf{G}_0^{2^\ell}$*.*

The proof of Theorem 3.8 relies on a result by Ahmadinejad et al. [1] saying that unit-circle approximations are preserved under arbitrary *true* powers.

▶ **Lemma 3.9** ([1] Corollary 4.9)**.** *Let* $\widetilde{\mathbf{G}}, \mathbf{G}$ *be directed multigraphs. If* $\widetilde{\mathbf{G}} \overset{\circ}{\approx}_{\varepsilon} \mathbf{G}$ *then for all* $k \in \mathbb{N}$ *we have* $\widetilde{\mathbf{G}}^k \overset{\circ}{\approx}_{\varepsilon/(1-\frac{3}{2}\varepsilon)} \mathbf{G}^k$*.*

Proof of Theorem 3.8. By Theorem 3.7, for all j, $\mathbf{G}_{i+1} \overset{\circ}{\approx}_{2\varepsilon} \mathbf{G}_i^2$. We then use Lemma 3.9 which states that we can take arbitrary powers and preserve unit-circle approximation. For arbitrary $i \in [\ell], k_i \in \mathbb{N}$ we have $\mathbf{G}_{i+1}^{k_i} \overset{\circ}{\approx}_{\frac{2\varepsilon}{1-3\varepsilon}} \mathbf{G}_i^{2k_i}$. Then by choosing $k_i = 2^{\ell-i}$ we obtain a chain

$$\mathbf{G}_\ell \overset{\circ}{\approx}_{\frac{2\varepsilon}{1-3\varepsilon}} \mathbf{G}_{\ell-1}^2 \overset{\circ}{\approx}_{\frac{2\varepsilon}{1-3\varepsilon}} \mathbf{G}_{\ell-2}^4 \overset{\circ}{\approx}_{\frac{2\varepsilon}{1-3\varepsilon}} \dots \overset{\circ}{\approx}_{\frac{2\varepsilon}{1-3\varepsilon}} \mathbf{G}_0^{2^\ell},$$

relating the final derandomized square to the true power via a sequence of unit-circle approximations. Applying Corollary 2.6 gives the bound $\mathbf{G}_\ell \overset{\circ}{\approx}_C \mathbf{G}_0^{2^\ell}$ where $C = 2\varepsilon\ell \cdot \frac{1-3\varepsilon}{(1-5\varepsilon)^2} \le 8\varepsilon\ell$. ◀

4 The Pseudorandom Generator

In this section, we present the PRG of Theorem 1.4. We first state the definition of the Impagliazzo-Nisan-Widgerson (INW) generator and relate it to the repeated derandomized square. For the remainder of the section, fix a sequence of c-regular undirected multigraphs $\mathbf{H}_1, \mathbf{H}_2, \dots$ where $\mathbf{H}_i$ has $d \cdot c^{i-1}$ vertices and has a one way labeling. We define a sequence of generators $\mathrm{INW}_0, \mathrm{INW}_1, \dots$ such that $\mathrm{INW}_i : [d] \times [c]^i \to [d]^{2^i}$.

▶ **Definition 4.1** (INW Generator [9])**.** *Define* $\mathrm{INW}_0(\sigma) = \sigma$ *for* $\sigma \in [d]$ *as the trivial PRG that outputs its input and* $\mathrm{INW}_{i+1}(v, e) = (\mathrm{INW}_i(v), \mathrm{INW}_i(\mathbf{H}_{i+1}[v, e]))$*.*

This is the recursive definition of the INW generator [9]. However, in the context of graphs with consistent one-way labelings there exists an equivalent characterization in terms of the derandomized square [20], which we will use for our analysis. The following two lemmas follow from the reasoning in Rozenman and Vadhan's work [20, Theorem 5.8]. We repeat the proofs here for completeness.

▶ **Lemma 4.2.** *Let $\mathbf{G}$ be a d-regular multigraph and $\mathbf{H}$ a c-regular undirected multigraph on d vertices. If $\mathbf{G}$ has a consistent one-way labeling, then $\overline{\mathbf{G}} \circledS \mathbf{H}$ has a consistent one-way labeling.*

Proof. Let $\widetilde{\mathbf{G}} = \overline{\mathbf{G}} \circledS \mathbf{H}$. By the definitions of $\overline{\mathbf{G}}$ and $\circledS$, we have

$$\widetilde{\mathbf{G}}[v,(i,j)] = \mathbf{G}[\mathbf{G}[v,i],\mathbf{H}[i,j]].$$

To prove that $\widetilde{\mathbf{G}}$ has a consistent one-way labeling, fix (i,j), and suppose $\widetilde{\mathbf{G}}[u,(i,j)] = \widetilde{\mathbf{G}}[v,(i,j)]$. We must show that $u = v$. Indeed, $\mathbf{G}[\mathbf{G}[u,i],\mathbf{H}[i,j]] = \mathbf{G}[\mathbf{G}[v,i],\mathbf{H}[i,j]]$. Since $\mathbf{G}$ has a consistent one-way labeling, this implies that $\mathbf{G}[u,i] = \mathbf{G}[v,i]$. Again using the fact that $\mathbf{G}$ has a consistent one-way labeling, this implies that $u = v$ as desired. ◀

▶ **Lemma 4.3.** *Let $\mathbf{G}_0$ be a d-regular multigraph on any number of vertices with a consistent one-way labeling. For $i \geq 0$, inductively define $\mathbf{G}_{i+1} = \overline{\mathbf{G}_i} \circledS \mathbf{H}_{i+1}$. Then for all v and e, $\mathbf{G}_i[v,e] = \mathbf{G}_0^{2^i}[v,\mathrm{INW}_i(e)]$.*

Proof. First, note that inductively, $\mathbf{G}_i$ has a consistent labeling by Lemma 4.2, so $\mathbf{G}_{i+1}$ is well-defined. Now we show by induction on i that $\mathbf{G}_i[v,e] = \mathbf{G}_0^{2^i}[v,\mathrm{INW}_i(e)]$. The case of $\mathbf{G}_0$ is immediate. Assume the inductive hypothesis holds for i. Fix an arbitrary vertex v and edge label $e = (e_1,e_2) \in [d \cdot c^i] \times [c]$. We have

$$\begin{aligned}
\mathbf{G}_{i+1}[v,e] &= \mathbf{G}_i[\mathbf{G}_i[v,e_1],\mathbf{H}_{i+1}[e_1,e_2]] && \text{(Definitions)}\\
&= \mathbf{G}_0^{2^i}[\mathbf{G}_0^{2^i}[v,\mathrm{INW}_i(e_1)],\mathrm{INW}_i(\mathbf{H}_{i+1}[e_1,e_2])] && \text{(Induction hypothesis)}\\
&= \mathbf{G}_0^{2^{i+1}}[v,(\mathrm{INW}_i(e_1),\mathrm{INW}_i(\mathbf{H}_{i+1}[e_1,e_2]))]\\
&= \mathbf{G}_0^{2^{i+1}}[v,\mathrm{INW}_{i+1}(e)].
\end{aligned}$$ ◀

Let $\ell = \lceil \log(n) \rceil$, and define $G\colon [d] \times [c]^\ell \to [d]^n$ by letting $G(x)$ be the n-symbol prefix of $\mathrm{INW}_\ell(x)$. This will be the generator that proves Theorem 1.4.

4.1 Approximation Guarantee

To bridge the gap between regular graphs and branching programs, we now define the *execution graph* of a branching program, which is just like the standard graph representation of the program, but the length is padded to a power of two and edges are added to wrap around from the end to the beginning.

▶ **Definition 4.4** (Branching Program Execution Graph)**.** *Let B be a permutation branching program of width w, degree d and length n, and let m be the smallest power of 2 greater than n. Define the **execution graph** of B to be a directed d-regular multigraph $\mathbf{G}$ on the vertex set $\{0,\ldots,m\} \times [w]$ with a one-way labeling given by*

$$\mathbf{G}[(t,u),\sigma] = \begin{cases} (t+1, W_t(u,\sigma)) & t \in \{0,\ldots,n-1\} \\ (t+1,u) & t \in \{n,\ldots,m-1\} \\ (0,u) & t = m, \end{cases}$$

where W_t is the transition function of B at layer i as in Definition 1.3.

▶ Remark 4.5. Since B is a permutation branching program, the execution graph $\mathbf{G}$ has a *consistent* one-way labeling. This is not true for general regular branching programs and is why our method does not generalize.

▷ Claim 4.6. Let $\varepsilon > 0$. If every $\mathbf{H}_i$ satisfies $\lambda(\mathbf{H}_i) \le \varepsilon/(8\ell)$, then G is an ε-PRG for permutation branching programs of degree d, length n, and arbitrary width.

Proof. Let B be an arbitrary permutation branching program of degree d and length n. Let $\mathbf{G}_0$ be the execution graph of B. Let $\mathbf{G}_1, \mathbf{G}_2, \ldots$ be the graphs in Lemma 4.3. By Theorem 3.8 we have $\mathbf{G}_\ell \overset{\circ}{\approx}_\varepsilon \mathbf{G}_0^m$.

Let $u = (0, v_{\text{start}})$ be the start vertex in the execution graph, and let $v = (m, v_{\text{end}})$ be the accept vertex. By the definition of $\mathbf{G}_0$, for all $\sigma \in [d]^m$, we have $\mathbf{G}_0^m[u,\sigma] = (m,a)$, where a is the final state of B when it reads $\sigma_{1\ldots n}$. Therefore, $\mathbf{G}_0^m[u,\sigma] = v \iff B(\sigma_{1\ldots n}) = 1$.

Let M be the random walk matrix of $\mathbf{G}_0$ and $\widetilde{M}$ the random walk matrix of $\mathbf{G}_\ell$. Then

$$\begin{aligned}
&\left| \Pr_{x \leftarrow U_{[d]\times[c]^\ell}}[B(G(x)) = 1] - \Pr_{\sigma \leftarrow U_{[d]^n}}[B(\sigma) = 1] \right| \\
&\quad = \left| \Pr_{x \leftarrow U_{[d]\times[c]^\ell}}[\mathbf{G}_0^m[u, \mathrm{INW}_\ell(x)] = v] - \Pr_{e \leftarrow U_{[d]^m}}[\mathbf{G}_0^m[u,e] = v] \right| \\
&\quad = \left| \Pr_{e \leftarrow U_{[d]\times[c]^\ell}}[\mathbf{G}_\ell[u,e] = v] - \Pr_{e \leftarrow U_{[d]^m}}[\mathbf{G}_0^m[u,e] = v] \right| \\
&\quad = \left| \widetilde{M}_{v,u} - M^m_{v,u} \right|,
\end{aligned}$$

which is at most ε by Proposition 2.7. ◁

To complete the proof of Theorem 1.4 we recall a result giving the existence of explicit expanders of all sizes.

▶ **Lemma 4.7** ([14] Theorem 3.3, Definition 2.13). *For all $n > 1$ and $\lambda > 0$, there is a $c = \mathrm{poly}(1/\lambda)$ and a c-regular undirected multigraph $\mathbf{H}$ on n vertices with a one-way labeling such that $\lambda(\mathbf{H}) \le \lambda$, and given λ, v, and e, the vertex $\mathbf{H}[v,e]$ can be computed in space $O(\log(nc))$.*

Proof of Theorem 1.4. Let $\mathbf{H}_i$ be the expander given by Lemma 4.7 with $\lambda = \varepsilon/(8\ell)$ and $n = d \cdot c^{i-1}$. This sequence $\mathbf{H}_1, \mathbf{H}_2, \ldots$ satisfies the requirements of Claim 4.6, so G constructed with this sequence is an ε-PRG. It remains to show the seed length and that the generator is explicit.

By construction the ℓth INW generator INW_ℓ has domain $[d] \times [c]^\ell$. By definition $\ell \le \log(n) + 1$ and by Lemma 4.7 the degree of $\mathbf{H}_i$ for all i is $c = \mathrm{poly}(\log(n)/\varepsilon)$, which gives a seed length of $s = O(\log d + \log n \cdot (\log\log n + \log(1/\varepsilon)))$.

Finally, G is explicit, in that the output of the generator can be computed in working space $O(s)$. This follows directly from Definition 4.1 and the explicitness of the expanders. ◀

4.2 Improved Seed Length for Tiny Error

So far, we have designed a PRG with seed length

$$O(\log d + \log n \cdot (\log\log n + \log(1/\varepsilon))). \tag{1}$$

In this section, we will present a simple reduction that yields an improved seed length when ε is extremely small.

▶ **Lemma 4.8.** *Suppose $G\colon \{0,1\}^s \to [d^m]^n$ is an ε-PRG for length-n degree-(d^m) permutation branching programs. Identify $[d^m] = [d]^m$, and think of G as a function $G\colon \{0,1\}^s \to [d]^{mn}$. Then G is an ε-PRG for length-(mn) degree-d permutation branching programs.*

Proof. Let B be a length-(mn) degree-d permutation branching program. Define a length-n degree-(d^m) branching program B' where one step of B' simulates m steps of B. Then B' is a permutation branching program, and B' computes the same function as B, so fooling B' implies fooling B. ◀

▶ **Corollary 4.9.** *For all $n, d \in \mathbb{N}$ and $\varepsilon > d^{-n/2}$, there is an explicitly computable ε-PRG $G\colon \{0,1\}^s \to [d]^n$ for permutation branching programs of length n, degree d, and arbitrary width. This PRG has seed length*

$$O\left(\log d + \log\left(\frac{n\log d}{\log(1/\varepsilon)}\right)\cdot(\log\log n + \log(1/\varepsilon))\right).$$

Proof. If $\log(1/\varepsilon) < \log d$, then the seed length of Equation 1 is already sufficient. Assume, therefore, that $\log(1/\varepsilon) \geq \log d$. Let $m = \left\lceil\frac{\log(1/\varepsilon)}{\log d}\right\rceil$ and let $n' = \lceil n/m\rceil$. Plugging into Equation 1, we have constructed already a PRG for length-n' degree-(d^m) permutation branching programs with seed length s, where

$$\begin{aligned}
s &\leq O(\log(d^m) + \log(n/m)\cdot(\log\log n + \log(1/\varepsilon))\\
&= O\left(\log(1/\varepsilon) + \log\left(\frac{n\log d}{\log(1/\varepsilon)}\right)\cdot(\log\log n + \log(1/\varepsilon))\right)\\
&= O\left(\log\left(\frac{n\log d}{\log(1/\varepsilon)}\right)\cdot(\log\log n + \log(1/\varepsilon))\right),
\end{aligned}$$

where the last step uses the assumption $\varepsilon \geq d^{-n/2}$ which implies $\log\left(\frac{n\log d}{\log(1/\varepsilon)}\right) \geq 1$. By Lemma 4.8, that same PRG fools length-(mn') degree-d permutation branching programs. Since $mn' > n$, by truncating to the first n symbols, we get the desired PRG for length-n degree-d permutation branching programs. ◀

5 A Random Function is Not a Good PRG

In this section, we prove that a random generator does not fool unbounded-width permutation branching programs, unless the seed length is $\Omega(n\log d)$. The proof is based on the following family of exponential-width permutation branching programs.

▶ **Lemma 5.1.** *Let n be a multiple of two, and let $\pi\colon [d]^{n/2} \to [d]^{n/2}$ be a permutation. There is a width-$(d^{n/2})$ length-n degree-d permutation branching program B such that*

$$B(x,y) = 1 \iff y = \pi(x).$$

Proof. Let $\mathbb{Z}_d$ denote the ring of integers modulo d. We identify the state space $[d^{n/2}]$ with the space $\mathbb{Z}_d^{n/2}$, a $\mathbb{Z}_d$-module. Let $e_1, \ldots, e_{n/2} \in \mathbb{Z}_d^{n/2}$ denote the standard "basis vectors," i.e., e_t has a 1 in coordinate t and 0 in all other coordinates. The transition function $W_t\colon \mathbb{Z}_d^{n/2} \times \mathbb{Z}_d \to \mathbb{Z}_d^{n/2}$ is given by

$$W_t(v,\sigma) = \begin{cases} v + \sigma\cdot e_t & \text{if } t \leq n/2\\ \pi^{-1}(\pi(v) - \sigma\cdot e_t) & \text{if } t > n/2.\end{cases}$$

These transition functions satisfy the permutation condition, because $W_t(W_t(v,\sigma), -\sigma) = v$. The start state of B is the zero element $0 \in \mathbb{Z}_d^{n/2}$, and the accepting state is $\pi^{-1}(0)$. By induction, when B reads an input $(x,y) \in (\mathbb{Z}_d^{n/2})^2$, it passes through the state x in layer $n/2$, and ultimately it arrives at the state $\pi^{-1}(\pi(x) - y)$ in the final layer. Thus, $B(x,y) = 1 \iff y = \pi(x)$. ◀

▶ **Theorem 5.2** (Failure of the Probabilistic Method)**.** *Let n be a multiple of 2. Let $s = \left\lfloor \frac{n \log d}{4} \right\rfloor - 1$, and sample a generator G uniformly at random from all functions $G\colon \{0,1\}^s \to [d]^n$. With probability at least $3/4$, there is some length-n degree-d permutation branching program B such that*

$$\left| \Pr_{\sigma \leftarrow U_{[d]^n}}[B(\sigma) = 1] - \Pr_{x \leftarrow U_{\{0,1\}^s}}[B(G(x)) = 1] \right| = 1 - d^{-n/2}.$$

Proof. Let $G_L, G_R\colon \{0,1\}^s \to [d]^{n/2}$ be the left and right halves of G respectively, i.e., $G(x) = G_L(x) \circ G_R(x)$. We claim that with high probability, G_L and G_R are both injective. Indeed, for each pair of distinct seeds $x, x' \in \{0,1\}^s$, the strings $G_L(x), G_L(x')$ are independent uniform $(n/2)$-symbol strings, so $\Pr_G[G_L(x) = G_L(x')] = d^{-n/2}$. The number of pairs (x, x') is at most $\binom{2^s}{2} \leq \frac{1}{2} 2^{2s} \leq 2^{-3} d^{n/2}$, where the last inequality is by our choice of s. Therefore, by the union bound, $\Pr_G[G_L \text{ is not injective}] \leq 2^{-3}$. The same argument applies to G_R as well, so except with probability $2 \cdot 2^{-3} = \frac{1}{4}$, G_L and G_R are both injective. In this case, there exists a permutation $\pi\colon [d]^{n/2} \to [d]^{n/2}$ such that for every seed x, $\pi(G_L(x)) = G_R(x)$. By Lemma 5.1, there is a length-n degree-d permutation branching program B such that $B(y, z) = 1 \iff z = \pi(y)$. Therefore, for every seed x, $B(G(x)) = 1$, so $\Pr_x[B(G(x)) = 1] = 1$. On the other hand, since π is a permutation, $\Pr_\sigma[B(\sigma) = 1] = d^{-n/2}$. ◀

6 Seed Length Lower Bound

In this section, we prove our lower bound on the seed length of any PRG for unbounded-width permutation branching programs, showing that our PRG's seed length is near-optimal. Except when ε is extremely small, the lower bound is $\Omega(\log d + \log n \cdot \log(1/\varepsilon))$.

▶ **Theorem 6.1.** *Let $d \geq 2$ and $n \geq 1$. Let $G\colon \{0,1\}^s \to [d]^n$ be an ε-PRG for length-n degree-d permutation branching programs of unbounded width, where $d^{-n/2} \leq \varepsilon \leq 0.49$. Then*

$$s \geq \Omega\left(\log d + \log\left(\frac{n \log d}{\log(1/\varepsilon)}\right) \cdot \log(1/\varepsilon)\right).$$

The proof of Theorem 6.1 is based on the same family of exponential-width branching programs that we used to prove Theorem 5.2. At an intuitive level, we argue that either the first half of the PRG's output is information-theoretically unpredictable given the second half, or vice versa. After all, if each half is somewhat predictable given the other half, there ought to exist a permutation π such that the pseudorandom string (x, y) has a noticeable chance (say at least 2ε) of satisfying $\pi(x) = y$, whereas a truly random string is extremely unlikely to satisfy $\pi(x) = y$. It follows that the PRG must use $\Omega(\log(1/\varepsilon))$ bits of seed above and beyond the seed length for sampling the first half or the second half individually.

To obtain a suitable permutation π, we rely on a version of the Kőnig-Egerváry theorem regarding maximum matchings in bipartite graphs. (In the lemma statement, think of p as a weight function on the edges of the complete bipartite graph $K_{N,N}$, and think of π as identifying a perfect matching.)

▶ **Lemma 6.2** (Kőnig-Egerváry theorem for fractional edge weights, [21, Theorem 17.1])**.** *For every integer $N \geq 1$ and every function $p\colon [N] \times [N] \to [0, \infty)$, there exist a permutation $\pi\colon [N] \to [N]$ and functions $q, r\colon [N] \to [0, \infty)$ such that*

$$\forall x, y \in [N],\ p(x, y) \leq q(x) + r(y) \tag{2}$$

and

$$\sum_{x \in [N]} p(x, \pi(x)) = \sum_{x \in [N]} q(x) + \sum_{y \in [N]} r(y). \tag{3}$$

Note that Equations 2 and 3 immediately imply that π maximizes $\sum_x p(x, \pi(x))$. A proof of Lemma 6.2 can be found in Schrijver's text [21, Theorem 17.1]. Alternatively, Lemma 6.2 follows from strong linear programming duality and the fact that the integer matching polytope equals the fractional matching polytope.

As outlined previously, we would now like to show that if each of X and Y is somewhat predictable given the other, then there is a noticeable chance that $\pi(X) = Y$. To rigorously formulate and prove this statement, we use the notion of Shannon entropy.

▶ **Definition 6.3.** *If X is a discrete random variable, the* entropy *of X is*

$$\mathrm{H}[X] = \mathop{\mathbb{E}}_{x \sim X}\left[\log\left(\frac{1}{\Pr[X = x]}\right)\right].$$

If X and Y are jointly distributed discrete random variables, the joint entropy $\mathrm{H}[X, Y]$ *is the entropy of the pair (X, Y), and the* conditional entropy *of X given Y is*

$$\mathrm{H}[X \mid Y] = \mathop{\mathbb{E}}_{y \sim Y}[\mathrm{H}[X \mid Y = y]] = \mathop{\mathbb{E}}_{\substack{x \sim X \\ y \sim Y}}\left[\log\left(\frac{1}{\Pr[X = x \mid Y = y]}\right)\right].$$

▶ **Lemma 6.4.** *Let $N \geq 1$, and let X and Y be jointly distributed random variables, each taking values in $[N]$. There exists a permutation $\pi \colon [N] \to [N]$ such that*

$$\Pr[\pi(X) = Y] \geq 2^{-\mathrm{H}[X|Y] - \mathrm{H}[Y|X]}.$$

Lemma 6.4 bears a resemblance to a well-known fact, which says that if we allow an *arbitrary* function π (not necessarily a permutation), the maximum possible value of $\Pr[\pi(X) = Y]$ is precisely $2^{-\mathrm{H}_\infty(Y|X)}$, where $\mathrm{H}_\infty(Y \mid X)$ is the "average min-entropy" of Y given X [7]. Our lemma is an interesting "symmetric" variant.

Proof. Let π, q, r be the functions guaranteed by Lemma 6.2 for the function $p(x, y) = \Pr[(X, Y) = (x, y)]$. Let $\mathcal{X} = \mathrm{Supp}(X)$ and $\mathcal{Y} = \mathrm{Supp}(Y)$. Then

$$\begin{aligned}
\Pr[\pi(X) = Y] &= \sum_{x \in [N]} q(x) + \sum_{y \in [N]} r(y) && \text{(Equation 3)} \\
&\geq \sum_{x \in \mathcal{X}} q(x) + \sum_{y \in \mathcal{Y}} r(y) \\
&= \sum_{\substack{x \in \mathcal{X} \\ y \in \mathcal{Y}}} \Pr[(X, Y) = (x, y)] \cdot \left(\frac{q(x)}{\Pr[X = x]} + \frac{r(y)}{\Pr[Y = y]}\right) \\
&\geq \sum_{\substack{x \in \mathcal{X} \\ y \in \mathcal{Y}}} \Pr[(X, Y) = (x, y)]^2 \cdot \frac{q(x) + r(x)}{\Pr[X = x] \cdot \Pr[Y = y]} \\
&\geq \sum_{\substack{x \in \mathcal{X} \\ y \in \mathcal{Y}}} \frac{\Pr[(X, Y) = (x, y)]^3}{\Pr[X = x] \cdot \Pr[Y = y]} && \text{(Equation 2)} \\
&= \mathop{\mathbb{E}}_{(x,y) \sim (X,Y)}[\Pr[X = x \mid Y = y] \cdot \Pr[Y = y \mid X = x]] \\
&\geq 2^{\mathbb{E}_{(x,y) \sim (X,Y)}[\log(\Pr[X = x | Y = y] \cdot \Pr[Y = y | X = x])]} && \text{(Jensen)} \\
&= 2^{-\mathrm{H}[X|Y] - \mathrm{H}[Y|X]}.
\end{aligned}$$

◀

To apply Lemma 6.4 to analyze pseudorandom distributions for permutation branching programs, we will use the standard *chain rule* for Shannon entropy.

▷ Claim 6.5 (Chain Rule). If X and Y are discrete random variables, then $\mathrm{H}[X, Y] = \mathrm{H}[X] + \mathrm{H}[Y \mid X]$.

▶ **Lemma 6.6.** *Let n be a multiple of two, let X and Y be random variables distributed over $[d]^{n/2}$, and let $\varepsilon \geq d^{-n/2}$. Assume that for every length-n degree-d permutation branching program B,*

$$|\Pr[B(U_{[d]^n}) = 1] - \Pr[B(X, Y) = 1]| \leq \varepsilon. \tag{4}$$

Then

$$\mathrm{H}[X, Y] \geq \frac{1}{2}\left(\mathrm{H}[X] + \mathrm{H}[Y] + \log\left(\frac{1}{2\varepsilon}\right)\right).$$

Proof. Let π be the permutation of Lemma 6.4. By Lemma 5.1, there is some length-n degree-d permutation branching program B such that $B(x, y) = 1 \iff \pi(x) = y$. Since π is a permutation, $\Pr[B(U_{[d]^n}) = 1] = d^{-n/2}$. Therefore, by Equation 4,

$$2^{-\mathrm{H}[X|Y] - \mathrm{H}[Y|X]} \leq d^{-n/2} + \varepsilon \leq 2\varepsilon.$$

Therefore,

$$\begin{aligned} \mathrm{H}[X, Y] &= \frac{1}{2}(\mathrm{H}[X] + \mathrm{H}[Y \mid X] + \mathrm{H}[Y] + \mathrm{H}[X \mid Y]) && \text{(Chain Rule)} \\ &\geq \frac{1}{2}\left(\mathrm{H}[X] + \mathrm{H}[Y] + \log\left(\frac{1}{2\varepsilon}\right)\right). \end{aligned}$$ ◀

To complete the proof of Theorem 6.1, we use the following standard fact about Shannon entropy.

▷ Claim 6.7. If X is a discrete random variable and f is a function, then $\mathrm{H}[f(X)] \leq \mathrm{H}[X]$.

Proof of Theorem 6.1. The seed length must be $\Omega(\log d)$ simply because the program can compute any arbitrary function of its first symbol. For $i \geq 0$, let

$$n_i = \left\lceil \frac{\log(1/\varepsilon)}{\log d} \right\rceil \cdot 2^i.$$

We will prove by induction on i that if a distribution X over $[d]^{n_i}$ fools length-n_i degree-d permutation branching programs with error ε, then

$$\mathrm{H}[X] \geq \frac{i}{2} \cdot \log\left(\frac{1}{2\varepsilon}\right).$$

The base case $i = 0$ is trivial. For the inductive step, consider a distribution (X, Y) over strings of length n_i, where $|X| = |Y| = n_{i-1}$. Since a permutation branching program can elect to ignore some of its input symbols, X and Y must each individually fool length-n_{i-1} degree-d permutation branching programs with error ε. Therefore, by induction,

$$\frac{1}{2}(\mathrm{H}[X] + \mathrm{H}[Y]) \geq \frac{(i-1)}{2} \cdot \log\left(\frac{1}{2\varepsilon}\right).$$

Furthermore, since $n_i \geq 2\log(1/\varepsilon)/\log d$, we have $\varepsilon \geq d^{-n_i/2}$, so we may apply Lemma 6.6 to complete the inductive step.

Now consider

$$i = \left\lfloor \log\left(\frac{n}{\lceil \log(1/\varepsilon)/\log d \rceil}\right) \right\rfloor .$$

Since $\varepsilon \geq d^{-n/2}$, we have $n/2 \leq n_i \leq n$. Let X be the truncation of $G(U_{\{0,1\}^s})$ to the first n_i symbols. Then

$$s = \mathrm{H}[U_{\{0,1\}^s}] \geq \mathrm{H}[G(U_{\{0,1\}^s})] \geq \mathrm{H}[X] \geq \frac{i}{2} \log\left(\frac{1}{2\varepsilon}\right),$$

where the first two inequalities follow from Claim 6.7. If $\log(1/\varepsilon) > \log d$, then $i = \Omega\left(\log\left(\frac{n \log d}{\log(1/\varepsilon)}\right)\right)$, so we are done. Meanwhile, if $\log d = \lambda \cdot \log(1/\varepsilon)$ for some $\lambda \geq 1$, then $i = \lfloor \log n \rfloor$, so we have shown

$$s \geq \frac{\lfloor \log n \rfloor}{2} \cdot \log\left(\frac{1}{2\varepsilon}\right).$$

We also have

$$s \geq \Omega(\log d) = \Omega(\lambda \cdot \log(1/\varepsilon)) \geq \Omega(\log \lambda \cdot \log(1/\varepsilon)).$$

Combining, we get

$$\begin{aligned} s &\geq \Omega\left((\log n + \log \lambda) \cdot \log(1/\varepsilon)\right) \\ &= \Omega\left(\log(n\lambda) \cdot \log(1/\varepsilon)\right) \\ &= \Omega\left(\log\left(\frac{n \log d}{\log(1/\varepsilon)}\right) \cdot \log(1/\varepsilon)\right). \end{aligned}$$ ◀

7 The Optimal Seed Length for Hitting Set Generators

Let $\mathcal{F}$ be a class of functions $B\colon [d]^n \to \{0,1\}$. Recall that an ε-*HSG* for $\mathcal{F}$ is a function $G\colon \{0,1\}^s \to [d]^n$ such that

$$\forall B \in \mathcal{F},\ \Pr[B(U_{[d]^n}) = 1] \geq \varepsilon \implies \exists x \in \{0,1\}^s, B(G(x)) = 1.$$

Thus, an HSG is a "one-sided" variant of a PRG.

In this section, we prove that any HSG for polynomial-width permutation branching programs is an HSG for unbounded-width permutation branching programs. As a corollary, we will show that the optimal (nonexplicit) seed length for an HSG for unbounded-width permutation branching programs is $O(\log(nd/\varepsilon))$.

▶ **Theorem 7.1.** *Let n be a positive integer, let $G\colon \{0,1\}^s \to [d]^n$ be a function, and let $\varepsilon > 0$.*

1. *There is a value $w = O(n/\varepsilon)$ such that if G is an $(\varepsilon/2)$-HSG for width-w length-n ordered branching programs, then G is an ε-HSG for unbounded-width length-n permutation branching programs.*
2. *There is a value $w = O(n^2/\varepsilon)$ such that if G is an $(\varepsilon/2)$-HSG for width-w length-n permutation branching programs, then G is an ε-HSG for unbounded-width length-n permutation branching programs.*

Item 2 is not necessary for the purpose of establishing the optimal seed length for HSGs for unbounded-width permutation branching programs. We include the proof because we find it interesting.

Proof. Let B be a length-n permutation branching program. We will define a function $f\colon [d]^n \to \{0,1\}$ such that $f(x) = 1 \implies B(x) = 1$ and $\Pr_{x \in [d]^n}[B(x) \neq f(x)] \leq \varepsilon/2$. Furthermore, we will show that f can be computed by an ordered branching program of width $O(n/\varepsilon)$, as well as by a permutation branching program of width $O(n^2/\varepsilon)$.

Think of B as a directed graph. Let $V_0, \dots, V_n$ be the layers of the graph. For each vertex v, let $p_{\to v}$ denote the probability that B passes through v when it reads a random input. Let $q = \lceil 2n/\varepsilon \rceil$. If the width of B is less than q, we can just let $f = B$, so assume the width of B is at least q. For each $t \in \{0, 1, \dots, n\}$, define S_t to be the set of q vertices $v \in V_t$ with the largest values of $p_{\to v}$. Let $f(x) = 1$ if the path through B described by x stays within $S_0, S_1, \dots, S_n$ and ends at the accepting vertex.

Clearly, $f(x) = 1 \implies B(x) = 1$. Now consider sampling $x = (x_1, \dots, x_n) \in [d]^n$ uniformly at random. For each $t \in [n]$ and each vertex $v \in V_t$, let $B_{v\to}$ denote the permutation branching program that ignores the first t symbols of its inputs and then simulates the last $(n - t)$ layers of B starting at vertex v. By the definition of S_t, each $v \in V_t \setminus S_t$ satisfies $p_{\to v} < 1/q$. Therefore,

$$\begin{aligned}
\Pr_x[B(x) \neq f(x)] &\leq \sum_{t=1}^{n} \sum_{v \in V_t \setminus S_t} p_{\to v} \cdot \Pr_x[B_{v\to}(x) = 1] \\
&< \frac{1}{q} \cdot \sum_{t=1}^{n} \sum_{v \in V_t \setminus S_t} \Pr_x[B_{v\to}(x) = 1] \\
&\leq \frac{1}{q} \cdot \sum_{t=1}^{n} \mathop{\mathbb{E}}_x\left[\sum_{v \in V_t} B_{v\to}(x)\right].
\end{aligned}$$

Consider any fixed t and x. By the permutation condition, it is possible to work backward from the accepting vertex to find the unique vertex $v \in V_t$ such that $B_{v\to}(x) = 1$. Therefore, $\sum_{v \in V_t} B_{v\to}(x) = 1$. Thus,

$$\Pr_x[B(x) \neq f(x)] \leq \frac{1}{q} \cdot \sum_{t=1}^{n} 1 \leq \frac{\varepsilon}{2}.$$

An ordered branching program for f can be obtained from B by deleting all the vertices in $V_t \setminus S_t$ and redirecting all their incoming edges to a new $\perp$ vertex. All outgoing edges from the $\perp$ vertex in layer t point to the $\perp$ vertex in layer $t + 1$, and finally in layer n, the $\perp$ vertex is a reject vertex. Clearly, the width of this program is $q + 1$.

Now let us define a permutation branching program computing f of width $w = q \cdot (n + 1)$. Let w_0 be the width of B, and number the states so that S_t corresponds to $[q] \subseteq [w_0]$. Let $W_t\colon [w_0] \times [d] \to [w_0]$ be the transition function of B. Let $A_{t,\sigma}$ be the set of $v \in [q]$ such that $W_t(v, \sigma) \in [q]$. By the permutation condition, for each fixed t and σ, the function $W_t(\cdot, \sigma)$ is a permutation on $[w_0]$. Therefore, there exists a permutation $\pi_{t,\sigma}\colon [w_0] \setminus A_{t,\sigma} \to [w_0] \setminus W_t(A_{t,\sigma}, \sigma)$.

Let $\mathbb{Z}_n$ denote the additive group of integers modulo n, and identify $[w] = [q] \times \mathbb{Z}_{n+1}$. The new branching program's transition function $W'_t\colon [q] \times \mathbb{Z}_{n+1} \times [d] \to [q] \times \mathbb{Z}_{n+1}$ is given by

$$W'_t(v, i, \sigma) = \begin{cases} (W_t(v, \sigma), i) & \text{if } v \in A_{t,\sigma} \\ (\pi_{t,\sigma}(v), i + 1) & \text{otherwise.} \end{cases}$$

Clearly, this satisfies the permutation condition. The start state is $(v_{\text{start}}, 0)$ and the accept state is $(v_{\text{end}}, 0)$, where v_{start} and v_{end} are the start and accept states of B. If $f(\sigma) = 1$, then inductively, when our permutation branching program reads σ, it simulates B and ultimately accepts without ever incrementing i. Conversely, if our permutation branching program accepts σ, then i must never be incremented. Therefore, when B reads σ, it stays within the sets $S_0, \ldots, S_n$ and accepts, and hence $f(\sigma) = 1$.

Finally, if $\Pr[B(U_{[d]^n}) = 1] \geq \varepsilon$, then $\Pr[f(U_{[d]^n}) = 1] \geq \varepsilon/2$. Therefore, under either of the two assumptions of the theorem, G hits f, and since $f \leq B$, this implies that G hits B as well. ◀

▶ **Corollary 7.2.** *For every n, d, ε, there exists an ε-HSG $G\colon \{0,1\}^s \to [d]^n$ for unbounded-width length-n degree-d permutation branching programs with seed length $s = O(\log(nd/\varepsilon))$.*

Proof. It is standard that there exists a nonexplicit ε-HSG for width-w length-n degree-d ordered branching programs with seed length $O(\log(wnd/\varepsilon))$. (Indeed, a random function is an HSG with these parameters with high probability.) ◀

The next claim shows that the seed length in Corollary 7.2 is optimal.

▷ Claim 7.3. Let $d \geq 2$ and $n \geq 1$. Let $G\colon \{0,1\}^s \to [d]^n$ be an ε-HSG for length-n degree-d permutation branching programs of unbounded width, where $d^{-n} \leq \varepsilon \leq 1/3$. Then $s \geq \Omega(\log(nd/\varepsilon))$.

Proof sketch. The seed length needs to be at least $\Omega(\log d)$ because the program can compute any function of the first input symbol. The seed length needs to be at least $\Omega(\log(1/\varepsilon))$ because unbounded-width permutation branching programs can check whether a prefix of the input is equal to a fixed arbitrary string. Finally, let $G\colon \{0,1\}^s \to [d]^n$ with $s < \log n$; we will show that G is not a $(1/3)$-HSG for degree-d permutation branching programs. Let $b\colon [d] \to \mathbb{F}_2$ be as close to balanced as possible. Since $2^s < n$, there is some nonzero vector $z \in \mathbb{F}_2^n$ such that for every seed x,

$$\sum_{i=1}^{n} z_i \cdot b(G(x)_i) = 0.$$

The function $B(x) = \sum_{i=1}^n z_i \cdot b(x_i)$ can be computed by a width-2 degree-d permutation branching program, and $\Pr_x[B(x) = 1] \geq 1/3$. ◁

8 Directions for Further Research

The obvious challenge is to obtain optimal PRGs for unbounded-width permutation branching programs in the large-error regime. We conjecture that our seed length lower bound is tight, i.e., there is a PRG construction that eliminates the $\log \log n$ factor from our PRG's seed length.

We showed that there is a nonexplicit HSG with seed length $O(\log(n/\varepsilon))$ for unbounded-width permutation branching programs. A natural problem is to match the seed length with an explicit construction. In the constant-width case, Braverman et al. [2] presented a simple HSG for the more general model of regular branching programs with seed length $O(\log n)$, independent of ε.

We wonder what PRGs can be constructed for the more challenging model of *arbitrary-order* permutation branching programs. Reingold, Steinke, and Vadhan [17] and Chattopadhyay et al. [3] have constructed PRGs for the small-width case. By using one generator for large ε and the other for small ε, one can achieve seed length $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$ when the width is a constant. For the unbounded-width case, explicit constructions or bounds for nonexplicit PRGs would be interesting.

Finally, we wonder whether our results can be generalized to the case of unbounded-width regular branching programs. Our HSG existence proof (Theorem 7.1 and Corollary 7.2) does generalize to the regular case[3], but the PRG situation is unclear.

References

1 AmirMahdi Ahmadinejad, Jonathan Kelner, Jack Murtagh, John Peebles, Aaron Sidford, and Salil Vadhan. High-precision estimation of random walks in small space. In *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2020. To appear.

2 Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. Pseudorandom generators for regular branching programs. *SIAM Journal on Computing*, 43(3):973–986, 2014. doi:10.1137/120875673.

3 Eshan Chattopadhyay, Pooya Hatami, Kaave Hosseini, and Shachar Lovett. Pseudorandom generators from polarizing random walks. *Theory of Computing. An Open Access Journal*, 15:Paper No. 10, 26, 2019. doi:10.4086/toc.2019.v015a010.

4 Michael B. Cohen, Jonathan Kelner, John Peebles, Richard Peng, Anup B. Rao, Aaron Sidford, and Adrian Vladu. Almost-linear-time algorithms for Markov chains and new spectral primitives for directed graphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 410–419. ACM, New York, 2017. doi:10.1145/3055399.3055463.

5 Anindya De. Pseudorandomness for permutation and regular branching programs. In *26th Annual IEEE Conference on Computational Complexity*, pages 221–231. IEEE Computer Soc., Los Alamitos, CA, 2011. doi:10.1109/CCC.2011.23.

6 Anindya De, Omid Etesami, Luca Trevisan, and Madhur Tulsiani. Improved pseudorandom generators for depth 2 circuits. In *Approximation, randomization, and combinatorial optimization*, volume 6302 of *Lecture Notes in Computer Science*, pages 504–517. Springer, Berlin, 2010. doi:10.1007/978-3-642-15369-3_38.

7 Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, 2008. doi:10.1137/060651380.

8 Shlomo Hoory and Avi Wigderson. Universal traversal sequences for expander graphs. *Information Processing Letters*, 46(2):67–69, 1993. doi:10.1016/0020-0190(93)90199-J.

9 Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC)*, page 356–364, New York, NY, USA, 1994. Association for Computing Machinery. doi:10.1145/195058.195190.

10 Michal Koucký, Prajakta Nimbhorkar, and Pavel Pudlák. Pseudorandom generators for group products. In *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC)*, pages 263–272. ACM, New York, 2011. doi:10.1145/1993636.1993672.

11 M. Luby and B. Veličković. On deterministic approximation of DNF. *Algorithmica*, 16(4-5):415–433, 1996. doi:10.1007/s004539900054.

12 Raghu Meka and David Zuckerman. Pseudorandom generators for polynomial threshold functions. *SIAM Journal on Computing*, 42(3):1275–1301, 2013. doi:10.1137/100811623.

13 Jack Murtagh, Omer Reingold, Aaron Sidford, and Salil Vadhan. Derandomization beyond connectivity: undirected Laplacian systems in nearly logarithmic space. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 801–812. IEEE Computer Soc., Los Alamitos, CA, 2017. doi:10.1109/FOCS.2017.79.

[3] In the regular case, we have $\mathbb{E}_x[\sum_{v\in V_t} B_{v\to}(x)] = \sum_{v\in V_t} \Pr_x[B_{v\to}(x) = 1] = 1$, where the last equality can be proven by backward induction on t.

14 Jack Murtagh, Omer Reingold, Aaron Sidford, and Salil Vadhan. Deterministic Approximation of Random Walks in Small Space. In Dimitris Achlioptas and László A. Végh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*, volume 145 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 42:1–42:22, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.42.

15 Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992. doi:10.1007/BF01305237.

16 Omer Reingold. Undirected connectivity in log-space. *Journal of the ACM*, 55(4):Art. 17, 24, 2008. doi:10.1145/1391289.1391291.

17 Omer Reingold, Thomas Steinke, and Salil Vadhan. Pseudorandomness for regular branching programs via Fourier analysis. In *Approximation, randomization, and combinatorial optimization*, volume 8096 of *Lecture Notes in Computer Science*, pages 655–670. Springer, Heidelberg, 2013. doi:10.1007/978-3-642-40328-6_45.

18 Omer Reingold, Luca Trevisan, and Salil Vadhan. Pseudorandom walks on regular digraphs and the **RL** vs. **L** problem. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 457–466. ACM, New York, 2006. doi:10.1145/1132516.1132583.

19 Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders. *Annals of Mathematics. Second Series*, 155(1):157–187, 2002. doi:10.2307/3062153.

20 Eyal Rozenman and Salil Vadhan. Derandomized squaring of graphs. In *Approximation, randomization and combinatorial optimization*, volume 3624 of *Lecture Notes in Computer Science*, pages 436–447. Springer, Berlin, 2005. doi:10.1007/11538462_37.

21 Alexander Schrijver. *Combinatorial optimization. Polyhedra and efficiency. Vol. A*, volume 24 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 2003. Paths, flows, matchings, Chapters 1–38.

22 Thomas Steinke. Pseudorandomness for permutation branching programs without the group theory. ECCC preprint TR12-083, 2012.