THEORY OF COMPUTING, Volume 22 (3), 2026, pp. 1–63
www.theoryofcomputing.org

SPECIAL ISSUE: CCC 2021

Pseudodistributions That Beat All Pseudorandom Generators

Edward Pyne*   Salil Vadhan†

Received July 22, 2021; Revised September 17, 2025; Published June 8, 2026

Abstract. A recent paper of Braverman, Cohen, and Garg (STOC'18) introduced the concept of a *weighted pseudorandom generator (WPRG)*, which amounts to a pseudorandom generator (PRG) whose outputs are accompanied with (potentially negative) real coefficients that scale the acceptance probabilities of any potential distinguisher. They gave an explicit construction of WPRGs for ordered branching programs whose seed length has a better dependence on the error parameter ε than the classic PRG construction of Nisan (STOC'90 and Combinatorica 1992).

In this article we give an explicit construction of WPRGs that achieve parameters that are *impossible* to achieve by a PRG. In particular, we construct a WPRG for *ordered permutation branching programs of unbounded width* with a single accept state that has seed length $\widetilde{O}(\log^{3/2} n)$ for error parameter $\varepsilon = 1/\text{poly}(n)$, where n is the input

An extended abstract of this paper appeared in the Proceedings of the 36th Computational Complexity Conference (CCC'21) [38].

*Supported by NSF grant CCF-1763299. The work was completed while the author was an undergraduate student at Harvard University.

†Supported by NSF grant CCF-1763299 and a Simons Investigator Award.

ACM Classification: F.1.3

AMS Classification: 68W20

Key words and phrases: pseudorandomness, space-bounded computation, spectral graph theory

DOI: 10.4086/toc.2026.v022a003

length. In contrast, recent work of Hoza et al. (ITCS'21) shows that any PRG for this model requires seed length $\Omega(\log^2 n)$ to achieve error $\varepsilon = 1/\mathrm{poly}(n)$.

As a corollary, we obtain explicit WPRGs with seed length $\widetilde{O}(\log^{3/2} n)$ and error $\varepsilon = 1/\mathrm{poly}(n)$ for ordered permutation branching programs of width $w = \mathrm{poly}(n)$ with an arbitrary number of accept states. Previously, seed length $o(\log^2 n)$ was only known when both the width and the reciprocal of the error are subpolynomial, i. e., $w = n^{o(1)}$ and $\varepsilon = 1/n^{o(1)}$ (Braverman, Rao, Raz, Yehudayoff, FOCS'10 and SICOMP 2014).

The starting point for our work was the recent family of space-efficient algorithms for estimating random-walk probabilities in directed graphs by Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan (FOCS'20), which are based on spectral graph theory and space-efficient Laplacian solvers. We interpret these algorithms as giving WPRGs with large seed length, which we then derandomize to obtain our results. We also note that this approach gives a simpler proof of the original result of Braverman, Cohen, and Garg, as independently discovered by Cohen, Doron, Renard, Sberlo, and Ta-Shma (CCC'21).

1 Introduction

The notion of a **pseudorandom generator (PRG)** [7, 46, 33, 35] is ubiquitous in theoretical computer science, with vast applicability in cryptography and derandomization. (See the texts [21, 45] for more background on pseudorandomness.) A recent paper by Braverman, Cohen, and Garg [10] introduced the following intriguing generalization of a PRG, in which we attach real coefficients to the outputs of the generator.

Definition 1.1. Let $\mathcal{B}$ be a class of boolean functions $B\colon \{0,1\}^n \to \{0,1\}$. An ε**-weighted pseudorandom generator (WPRG)** for $\mathcal{B}$ is a function $(G,\rho)\colon \{0,1\}^s \to \{0,1\}^n \times \mathbb{R}$ such that for every $B \in \mathcal{B}$,

$$\left| \mathop{\mathbb{E}}_{x \leftarrow U_n}[B(x)] - \mathop{\mathbb{E}}_{x \leftarrow U_s}[\rho(x) \cdot B(G(x))] \right| \le \varepsilon.$$

The value s is the **seed length** of the WPRG, and n is the **output length** of the WPRG. We say that the WPRG is **(mildly)[1] explicit** if given x, $G(x)$ and $\rho(x)$ are computable in space $O(s)$, and $\rho(x)$ has absolute value at most $2^{O(s)}$.[2]

Above and throughout, we use the standard definition of space-bounded complexity, which counts the working, read-write memory of the algorithm, and does not include the length of the read-only input or write-only output, which can be exponentially longer than the space bound.

[1] We consider this definition to correspond to *mild* explicitness because requiring that the generator be computable in space linear in its seed length only implies that it is computable in time exponential in its seed length (i. e., time polynomial in the size of its truth table), which is mildly explicit according to the terminology in [45]. *Strong* explicitness, in contrast, would require that each bit of the truth table be computable in time polynomial in s.

[2] Naively the coordinates could be as large as $2^{2^{O(s)}}$, but this restriction is essentially WLOG for most models (Section 8).

In the original article by Braverman, Cohen, and Garg [10] and previous versions of this paper [37], generators as above were called **pseudorandom pseudodistributions (PRPDs)**. The terminology of weighted pseudorandom generators (WPRGs) was introduced by Cohen et al. [17], and we find it more intuitive (and it avoids the double use of the 'pseudo-' prefix).

With Definition 1.1, a PRG is a special case of a WPRG with $\rho(x) = 1$ for all x. The power of WPRGs comes from allowing the coefficients to be negative, which yields cancellations. Indeed, an explicit ε-WPRG with seed length s in which all of the coefficients are nonnegative can be converted into an explicit $O(\varepsilon)$-PRG with seed length $s + O(\log(1/\varepsilon))$ (see Section 8). A WPRG for a class of functions that includes testing if the input equals an arbitrary fixed string must have coefficients bounded by $2^{O(s)}$ (see Section 8).

A general WPRG can be converted into a linear combination of two unweighted generators. That is, for every explicit WPRG $(G, \rho)\colon \{0,1\}^s \to \{0,1\}^n \times \mathbb{R}$, there are explicit generators $G_+ : \{0,1\}^{s'} \to \{0,1\}^n$ and $G_- : \{0,1\}^{s'} \to \{0,1\}^n$ with seed length $s' = O(s + \log(1/\varepsilon))$ and (explicitly computable) coefficients $\rho_+, \rho_- \in \mathbb{R}^{\geq 0}$ such that for every function $B : \{0,1\}^n \to \{0,1\}$, we have

$$\mathbb{E}_x[\rho(x) \cdot B(G(x))] = \rho_+ \cdot \mathbb{E}_x[B(G_+(x))] - \rho_- \cdot \mathbb{E}_x[B(G_-(x))] \pm \varepsilon.$$

(See Section 8).

The motivation for WPRGs is that they can be used to derandomize algorithms in the same way as a PRG: we can estimate the acceptance probability of any function $B \in \mathcal{B}$ by enumerating over the seeds x of the WPRG (G, ρ) and calculating the average of the values $\rho(x) \cdot B(G(x))$. Furthermore, [10] observe that if (G, ρ) is an ε-WPRG then G is an ε-**hitting set generator (HSG)** for the same class. That is, if B is any function in $\mathcal{B}$ with $\Pr[B(U_n) = 1] > \varepsilon$, then there exists an $x \in \{0,1\}^s$ such that $B(G(x)) = 1$.

Given this motivation, it is natural to ask whether WPRGs are more powerful than PRGs. That is, can WPRGs achieve a shorter seed length than PRGs for a natural computational model $\mathcal{B}$? (There are simple constructions of artificial examples, one of which we give in Section 8.) As discussed below, Braverman, Cohen, and Garg [10] gave an explicit construction of WPRGs achieving a shorter seed length than the *best known* construction of PRGs for ordered branching programs, but not beating the best possible seed length for that model (given by a non-explicit application of the Probabilistic Method). In this paper, we give an explicit construction of WPRGs for a natural computational model (ordered permutation branching programs of unbounded width) with a seed length that beats all possible PRGs for that model.

1.1 Ordered branching programs

The work of Braverman, Cohen, and Garg [10], as well as our paper, focuses on WPRGs for classes $\mathcal{B}$ of functions computable by *ordered branching programs*, a nonuniform model that captures how a space-bounded randomized algorithm accesses its random bits.

Definition 1.2. An **(ordered) branching program** B of length n and width w computes a function $B : \{0,1\}^n \to \{0,1\}$. On an input $\sigma \in \{0,1\}^n$, the branching program computes as follows. It starts at a fixed start state $v_0 \in [w]$. Then for $r = 1, \ldots, n$, it reads the next symbol σ_r and updates its state according to a transition function $B_r : [w] \times \{0,1\} \to [w]$ by taking $v_t = B_r(v_{t-1}, \sigma_t)$. Note that the transition function B_r can differ at each time step.

The branching program **accepts** σ, denoted $B(\sigma) = 1$, if $v_n \in V_{\text{acc}}$, where $V_{\text{acc}} \subseteq [w]$ is the set of accept states, and otherwise it **rejects**, denoted $B(\sigma) = 0$. Thus an ordered branching program is specified by the transition functions $B_1, \dots, B_n$, the start state v_0 and the set V_{acc} of accept states.

An ordered branching program is necessarily read-once, where the read order is fixed in advance. An ordered branching program of length n and width $O(w)$ 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 and configuration of the Turing machine at that time. We note that we can convert any ordered branching program into one with a single accept state by collapsing all of V_{acc} to a single state.

Using the Probabilistic Method, it can be shown that there *exists* an ε-PRG for ordered branching programs of length n and width w with seed length $s = O(\log(nw/\varepsilon))$. The classic construction of Nisan [34] gives an explicit PRG with seed length $s = O(\log n \cdot \log(nw/\varepsilon))$, and this bound has not been improved except for extreme ranges of w, namely when w is at least quasipolynomially larger than n/ε [36, 5, 29] or when $w \leq 3$ [8, 43, 24, 32]. Braverman, Cohen, and Garg [10] gave an explicit construction of a WPRG that achieves improved dependence on the error parameter ε, with seed length

$$s = \widetilde{O}\left(\log n \cdot \log(nw) + \log(1/\varepsilon)\right).$$

In particular, for error $\varepsilon = n^{-\log n}$ and width $w = \text{poly}(n)$, their seed length improves Nisan's from $O(\log^3 n)$ to $\widetilde{O}(\log^2 n)$. Chattopadhyay and Liao [13] gave a simpler construction of WPRGs with a slightly shorter seed length than [10], with an additive dependence on $O(\log(1/\varepsilon))$ rather than $\widetilde{O}(\log(1/\varepsilon))$.

1.2 Permutation branching programs

Due to the lack of progress in constructing improved PRGs for general ordered branching programs as well as some applications [28, 30], attention has turned to more restricted classes of ordered branching programs. In this article, our focus is on *permutation* branching programs.

Definition 1.3. An **(ordered) permutation branching program** is an ordered branching program B where for all $t \in [n]$ and $\sigma \in \{0,1\}$, $B_t(\cdot, \sigma)$ is a permutation on $[w]$.

This can be thought of as the computation being time-reversible on any fixed input σ. We note that we cannot assume without loss of generality that a permutation branching program has a single accept state, as merging a set of accept states will destroy the permutation property. Nevertheless, ordered permutation branching programs with a single accept state can compute interesting functions, such as testing whether a $\sum_{i \in S} x_i \equiv 0 \pmod{m}$, for any $m \leq w$ and any $S \subseteq [n]$. An ordered permutation branching program with a single accept state can also compute permutation test functions, which test whether $x|_T = \pi(x|_S)$ for any permutation

$\pi : \{0,1\}^\ell \to \{0,1\}^\ell$ and any two subsets $S, T \subseteq [n]$ of size ℓ such that all elements of T are larger than all elements of S, provided that $w \geq 2^\ell$ [26].

Previous work on various types of PRGs for permutation branching programs [41, 40, 11, 31, 20, 44, 26] has achieved seed lengths that are logarithmic or nearly logarithmic in the length n of the branching program, improving the $\log^2 n$ bound in Nisan's generator. In particular, Braverman, Rao, Raz, and Yehudayoff [11] gave a PRG for the more general model of *regular* branching programs (with an arbitrary number of accept states) with seed length

$$s = O\left(\log n \cdot (\log w + \log(1/\varepsilon) + \log\log n)\right).$$

For getting an HSG, they also showed how to eliminate the $\log\log n$ and $\log(1/\varepsilon)$ terms at the price of a worse dependence on w,[3] achieving a seed length of

$$s \leq \log(n+1) \cdot w.$$

For the specific case of permutation branching programs, Koucký, Nimbhorkar, and Pudlák [31], De [20], and Steinke [44] showed how to remove the $\log\log n$ term in the Braverman et al. PRG at the price of a worse dependence on w, achieving seed length

$$s = O(\log n \cdot (\mathrm{poly}(w) + \log(1/\varepsilon))).$$

Most recently, Hoza, Pyne, and Vadhan [26] showed that the dependence on the width w could be entirely eliminated if we restrict to permutation branching programs with a *single accept state*, constructing a PRG with seed length

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

In particular, they show that this seed length is provably better than what is achieved by the Probabilistic Method; that is, a random function with seed length $o(n)$ fails to be a PRG for unbounded-width permutation branching programs with high probability. To do so, they use the family of permutation test programs discussed above. Like the prior PRGs for bounded-width permutation branching programs, the seed length has a term of $O(\log n \cdot \log(1/\varepsilon))$. However, in contrast to the bounded-width case, this cannot be improved to $O(\log(n/\varepsilon))$ by a non-explicit construction. Hoza et al. prove that seed length $\Omega(\log n \cdot \log(1/\varepsilon))$ is *necessary* for any ε-PRG against unbounded-width permutation branching programs. For hitting-set generators (HSGs), they show that seed length $O(\log(n/\varepsilon))$ is possible via the Probabilistic Method, thus leaving an explicit construction as an open problem.

1.3 Our results

In this paper, we construct an explicit WPRG for permutation branching programs of unbounded width and a single accept state that beats the aforementioned lower bounds for PRGs.

[3]The lack of dependence on ε can be explained by the observation of Braverman et al. that any regular branching program that has nonzero acceptance probability has acceptance probability at least $1/2^{w-1}$, so WLOG $\varepsilon > 1/2^w$, i. e., $w > \log(1/\varepsilon)$.

Theorem 1.4. *For all $n \in \mathbb{N}$ and $\varepsilon \in (0, 1/2)$, there is an explicit ε-WPRG (and hence ε-HSG) for ordered permutation branching programs of length n, arbitrary width, and a single accept state, with seed length*

$$s = O\left(\log(n)\sqrt{\log(n/\varepsilon)}\sqrt{\log\log(n/\varepsilon)} + \log(1/\varepsilon)\log\log(n/\varepsilon)\right).$$

In particular, when $\varepsilon = 1/\text{poly}(n)$, we achieve seed length $\widetilde{O}(\log^{3/2} n)$, while a PRG requires seed length $\Omega(\log^2 n)$ [26].

As noted in [26], an ε-WPRG for branching programs with a single accept state is also an $(a \cdot \varepsilon)$-WPRG for branching programs with at most a accept states. For bounded-width permutation branching programs, we can take $a = w$ and obtain the following.

Corollary 1.5. *For all $n, w \in \mathbb{N}$ and $\varepsilon \in (0, 1/2)$, there is an explicit ε-WPRG (and hence ε-HSG) for ordered permutation branching programs of length n and width w (and any number of accept states), with seed length*

$$s = O\left(\log(n)\sqrt{\log(nw/\varepsilon)}\sqrt{\log\log(nw/\varepsilon)} + \log(w/\varepsilon)\log\log(nw/\varepsilon)\right).$$

In particular for $w = \text{poly}(n)$ and $\varepsilon = 1/\text{poly}(n)$, we achieve seed length $\widetilde{O}(\log^{3/2} n)$. Note that the previous explicit PRGs (or even HSGs) for permutation branching programs (as mentioned in Subsection 1.2) achieved seed length $o(\log^2 n)$ only when both $w = n^{o(1)}$ and $\varepsilon = 1/n^{o(1)}$. With seed length $o(\log^2 n)$, Corollary 1.5 can handle width as large as $w = n^{\widetilde{\Omega}(\log n)}$ and error as small as $\varepsilon = 1/n^{-\widetilde{\Omega}(\log(n))}$. We summarize these results in a table.

Citation	Type	Model	Seed Length
Non-explicit (folklore)	PRG	General	$\Theta(\log(nw/\varepsilon)$
[34, 27]	PRG	General	$O(\log n \cdot \log(nw/\varepsilon))$
[11]	PRG	Regular	$\widetilde{O}(\log n \cdot \log(w/\varepsilon))$
[11]	HSG	Regular	$\log(n+1) \cdot w$
[31, 20, 44]	PRG	Permutation	$O(\log n \cdot (\text{poly}(w) + \log(1/\varepsilon))$
[10, 13], Thm. 4.1	WPRG	General	$\widetilde{O}(\log n \cdot \log nw + \log(1/\varepsilon))$
[26]	PRG	Permutation (1 accept)	$\widetilde{\Theta}(\log n \cdot \log(1/\varepsilon))$
Non-explicit [26]	HSG	Permutation (1 accept)	$O(\log(n/\varepsilon))$
Theorem 1.4	WPRG	Permutation (1 accept)	$\widetilde{O}(\log n\sqrt{\log(n/\varepsilon)} + \log(1/\varepsilon))$
Corollary 1.5	WPRG	Permutation	$\widetilde{O}(\log n\sqrt{\log(nw/\varepsilon)} + \log(w/\varepsilon))$

1.4 Subsequent work

Subsequent to the dissemination of the preprint version of this paper as ECCC TR21-019, there have been several further developments. Independently and concurrently, Cohen, Doron, Renard, Sberlo, and Ta-Shma [17] obtained Theorem 4.1, and both papers appeared at the 2021 Computational Complexity Conference. Subsequent to both preprints appearing, Hoza [25] combined Theorem 4.1 with the Armoni PRG [5] and the framework of Saks and Zhou [42] to slightly improve the derandomization of **BPL** to $\mathbf{BPSPACE}(S) \subset \mathbf{DSPACE}(S^{3/2}/\sqrt{\log S})$. He also constructed a weighted PRG for ordered branching programs with seed length $O(\log(n)\log(nw) + \log(1/\varepsilon))$, obtaining a weighted PRG with matching dependence on n and w to [34] with optimal dependence on ε. Subsequently, Bogdanov, Hoza, Prakriya, and Pyne [9] obtained a hitting set for regular branching programs with seed length $O(\log^{3/2} n + \log(n)\log(w) + \log(n)\sqrt{\log(1/\varepsilon)})$, improving on prior work [11] in the case that $\varepsilon = 1/\mathrm{poly}(n)$ and $w = n^{o(1)}$. Furthermore, Chen, Hoza, Lyu, Tal, and Wu [15] gave a new proof of Theorem 1.4. Their new proof could be extended to regular branching programs, allowing them to construct a weighted PRG for regular branching programs with seed length matching that of the hitting set of [9]. Finally, Chattopadhyay and Liao [14] substantially simplified the proof of [15] in the case of permutation programs. All of these works use the core idea of analyzing an error reduction procedure as a weighted PRG.

1.5 Versions of this paper

The original ECCC preprint version of this paper contained the main result of Theorem 1.4. The CCC publication was an extended abstract that did not include detailed proofs of the main claims. The present version contains proofs of the main claims, and a new section (Section 8) with some ancillary observations about weighted PRGs (this section also appeared in an updated version of the paper posted to ECCC).

2 Overview of proofs

The starting point for our work was the recent family of space-efficient algorithms for estimating random-walk probabilities in directed graphs by Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan [2], based on spectral graph theory and space-efficient Laplacian solvers. We interpret these algorithms as giving WPRGs with large seed length, which we then derandomize to obtain our results.

The specific problem considered by Ahmadinejad et al. is the following: given a directed graph $\mathcal{G} = (V, E)$, two vertices $s, t \in V$, a walk-length $k \in \mathbb{N}$, and an error parameter $\varepsilon > 0$, estimate the probability that a random walk of length k started at s ends at t to within $\pm\varepsilon$. Such an algorithm can be applied to the following graph in order to estimate the acceptance probability of an ordered branching program:

Definition 2.1. Given a length-n, width-w branching program B with transition functions $(B_1, \ldots, B_n)$ with start vertex $v_0 \in [w]$, and a single accept vertex v_{acc}, the **(layered) graph associated with** B is the graph $\mathcal{B}$ with vertex set $\{0, 1, \ldots, n\} \times [w]$ and directed edges from $(i-1, v)$ to $(i, B_i(v, 0))$ and $(i, B_i(v, 1))$ for every $i = 1, \ldots, n$ and $v \in [w]$.

Applying the algorithms of Ahmadinejad et al. to the graph $\mathcal{G}$ with $s = (0, v_0)$, $t = (n, v_{\mathrm{acc}})$, and $k = n$, we obtain an estimate of the acceptance probability of B to within $\pm\varepsilon$, just like an ε-WPRG for B would allow us to obtain. But a WPRG (G, ρ) is much more constrained than an arbitrary space-efficient algorithm, which can directly inspect the graph. Instead, a WPRG is limited to generating $S = 2^s$ walks of length n in the layered graph, described by sequences $G(x_1), \ldots, G(x_S) \in \{0,1\}^n$ of edge labels, and then combining the indicators $B(G(x_1)), \ldots, B(G(x_S))$ of whether the walks ended at t via a linear combination with fixed coefficients $\rho(x_1), \ldots, \rho(x_S) \in \mathbb{R}$.

Note that if B is a permutation branching program, then the graph $\mathcal{G}$ above is 2-regular (in that every vertex has 2 in- and out- edges, except for layer 0 which has no incoming edges and layer n which has no outgoing edges). Thus, the basis for Theorem 1.4 is the (main) result of Ahmadinejad et al., which applies to regular (or more generally, Eulerian) directed graphs G. However, they also give a new algorithm for estimating random-walk probabilities in arbitrary directed graphs. This algorithm is not as space-efficient as the ones for regular graphs, but is significantly simpler, so we begin by describing how to obtain a WPRG based on that algorithm. The resulting WPRG matches the parameters of the WPRG of Braverman, Cohen, and Garg [10], but has a significantly simpler proof (and is also simpler than the construction of Chattopadhyay and Liao [13]). A similar construction was independently discovered by Cohen, Doron, Renard, Sberlo, and Ta-Shma [17].

2.1 WPRG for arbitrary ordered branching programs

Let B be an arbitrary width-w, length-n ordered branching program, with associated layered graph $\mathcal{G}$ as in Definition 2.1. The algorithm of Ahmadinejad et al. starts with the $(n+1)w \times (n+1)w$ random-walk transition matrix $\mathbf{W}$ of $\mathcal{G}$, which has the following block structure

$$\mathbf{W} = \begin{bmatrix} 0 & \mathbf{B}_1 & 0 & \cdots & 0 \\ 0 & 0 & \mathbf{B}_2 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \ddots & \mathbf{B}_n \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Here entry $((i,u),(j,v))$ is the probability that taking one random step in $\mathcal{G}$ from vertex (i,u) ends at (j,v). Thus $\mathbf{B}_i$ is the $w \times w$ transition matrix for the random walk from layer $i-1$ to i in the branching program. (Note that the matrix $\mathbf{W}$ is not quite stochastic due to layer n having no outgoing edges.)

Ahmadinejad et al. consider the Laplacian $\mathbf{L} = \mathbf{I}_{(n+1)w} - \mathbf{W}$. Its inverse $\mathbf{L}^{-1} = (\mathbf{I}_{(n+1)w} - \mathbf{W})^{-1} = \mathbf{I}_{(n+1)w} + \mathbf{W} + \mathbf{W}^2 + \mathbf{W}^3 + \cdots$ sums up random-walks of all lengths in G, and thus has the following form.

$$\mathbf{L}^{-1} = \begin{bmatrix} \mathbf{B}_{0\ldots0} & \mathbf{B}_{0\ldots1} & \mathbf{B}_{0\ldots2} & \cdots & \mathbf{B}_{0\ldots n} \\ 0 & \mathbf{B}_{1\ldots1} & \mathbf{B}_{1\ldots2} & \cdots & \mathbf{B}_{1\ldots n} \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \ddots & \mathbf{B}_{n-1\ldots n} \\ 0 & 0 & 0 & \cdots & \mathbf{B}_{n\ldots n} \end{bmatrix},$$

where $\mathbf{B}_{i,i} = \mathbf{I}$ and

$$\mathbf{B}_{i..j} = \mathbf{B}_{i+1}\mathbf{B}_{i+2}\cdots\mathbf{B}_j.$$

In particular, the $(0,n)$'th block of $\mathbf{L}^{-1}$ gives the random-walk probabilities from layer 0 to layer n, and thus the acceptance probability of G is exactly the (v_0, v_{acc})'th entry of the $(0,n)$'th block of $\mathbf{L}^{-1}$. Therefore, the task reduces to producing a sufficiently good estimate of $\mathbf{L}^{-1}$.

Ahmadinejad et al. estimate $\mathbf{L}^{-1}$ in two steps. First, they observe that the Saks–Zhou derandomization of logspace [42] can be used to produce, in deterministic space $O(\log(nw)\sqrt{\log(n)})$, approximations $\widetilde{\mathbf{B}_{i..j}}$ of the blocks $\mathbf{B}_{i..j}$ to within entrywise error $1/\mathrm{poly}(nw)$, resulting in an approximate pseudoinverse

$$\widetilde{\mathbf{L}^{-1}} = \begin{bmatrix} \widetilde{\mathbf{B}_{0\ldots0}} & \widetilde{\mathbf{B}_{0\ldots1}} & \widetilde{\mathbf{B}_{0\ldots2}} & \cdots & \widetilde{\mathbf{B}_{0\ldots n}} \\ 0 & \widetilde{\mathbf{B}_{1\ldots1}} & \widetilde{\mathbf{B}_{1\ldots2}} & \cdots & \widetilde{\mathbf{B}_{1\ldots n}} \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \ddots & \widetilde{\mathbf{B}_{n-1\ldots n}} \\ 0 & 0 & 0 & \cdots & \widetilde{\mathbf{B}_{n\ldots n}} \end{bmatrix}, \tag{2.1}$$

with the property that

$$\left\|\mathbf{I}_{(n+1)w} - \widetilde{\mathbf{L}^{-1}}\mathbf{L}\right\|_1 \leq 1/nw,$$

where $\|\cdot\|_1$ denotes the ℓ_1 operator norm on row vectors, i. e., $\|\mathbf{M}\|_1 = \sup_{x\neq0}\|x\mathbf{M}\|_1/\|x\|_1$.

Next, Ahmadinejad et al. reduce the approximation error to an arbitrary $\varepsilon < 1/(nw)^{O(1)}$ by using preconditioned Richardson iterations, as captured by the following lemma.

Lemma 2.2 (preconditioned Richardson iteration, [2] Lemma 6.2). *Let $\|\cdot\|$ be a submultiplicative norm on $N \times N$ real matrices. Given matrices $\mathbf{A}, \mathbf{P}_0 \in \mathbb{R}^{N\times N}$ such that $\|\mathbf{I}_N - \mathbf{P}_0\mathbf{A}\| \leq \alpha$ for some $\alpha > 0$, let $\mathbf{P}_m = \sum_{i=0}^{m}(\mathbf{I}_N - \mathbf{P}_0\mathbf{A})^i\mathbf{P}_0$. Then $\|\mathbf{I}_N - \mathbf{P}_m\mathbf{A}\| \leq \alpha^{m+1}$.*

Setting $N = (n+1)w$, $\mathbf{A} = \mathbf{L}$, $\mathbf{P}_0 = \widetilde{\mathbf{L}^{-1}}$, and $\alpha = 1/n$, and $m = O(\log_n(1/\varepsilon))$, we obtain $\widetilde{\mathbf{L}_\varepsilon} = \mathbf{P}_m$ such that $\|\mathbf{I}_N - \widetilde{\mathbf{L}_\varepsilon}\mathbf{L}\|_1 \leq \varepsilon/(nw)^{O(1)}$, which implies that $\widetilde{\mathbf{L}_\varepsilon}$ and $\mathbf{L}^{-1}$ are entrywise equal up to $\pm\varepsilon$, for

$$\widetilde{\mathbf{L}_\varepsilon} = \sum_{i=0}^{m}(\mathbf{I}_N - \widetilde{\mathbf{L}^{-1}}\mathbf{L})^i\widetilde{\mathbf{L}^{-1}} \tag{2.2}$$

In particular, the (v_0, v_{acc})'th entry of the $(0,n)$'th block of $\widetilde{\mathbf{L}_\varepsilon}$ is an estimate of the acceptance probability of the branching program to within $\pm\varepsilon$. Computing $\widetilde{\mathbf{L}_\varepsilon}$ from $\mathbf{L}$ and $\widetilde{\mathbf{L}^{-1}}$ can be done in space $O((\log nw)\cdot\log m)$, yielding Ahmadinejad et al.'s space bound of

$$O(\log(nw)\sqrt{\log(n)} + (\log nw)\cdot\log\log_n(1/\varepsilon)).$$

Now we show how, with an appropriate modification, we can interpret this algorithm of Ahmadinejad et al. as a WPRG (albeit with large seed length). We replace the use of the Saks–Zhou algorithm (which requires looking at the branching program) with Nisan's pseudorandom generator. Specifically, we take $\widetilde{\mathbf{B}_{i..j}}$ to be the matrix whose (u, v)'th entry is the probability that, if we start at state u in the i'th layer and use a random output of Nisan's pseudorandom generator to take $j - i$ steps in the branching program, we end at state v in the j'th layer. For $\widetilde{\mathbf{B}_{i..j}}$ to approximate $\mathbf{B}_{i..j}$ to within error $\pm 1/\mathrm{poly}(nw)$ as above, Nisan's pseudorandom generator requires seed length

$$s_{\mathrm{Nisan}} = O(\log(j - i) \cdot \log nw) = O(\log n \cdot \log nw).$$

Observe that for every i, $\widetilde{\mathbf{B}_{i...i}} = \mathbf{I}_w = \mathbf{B}_{i...i}$. Without loss of generality, we may also assume that $\widetilde{\mathbf{B}_{(i-1)...i}} = \mathbf{B}_{(i-1)...i}$, since taking one step only requires one random bit.

Next, we observe from Equation (2.2) that the matrix $\widetilde{\mathbf{L}_\varepsilon}$ is a polynomial of degree $2m + 1$ in the matrices $\mathbf{L}$ and $\widetilde{\mathbf{L}^{-1}}$. In particular the $(0, n)$'th block of $\widetilde{\mathbf{L}_\varepsilon}$ is a polynomial of degree at most $2m + 1$ in the matrices $\widetilde{\mathbf{B}_{i..j}}$. Specifically, using the upper-triangular structure of the matrices $\mathbf{L}$ and $\widetilde{\mathbf{L}^{-1}}$ and noting that the product of d $(n + 1) \times (n + 1)$ block matrices expands into a sum of $(n + 1)^{d-1}$ terms, each of which is a product of d individual blocks, we show the following.

Observation 2.3. *The $(0, n)$'th block of $\widetilde{\mathbf{L}_\varepsilon}$ equals the sum of at most $(n + 1)^{O(m)}$ terms, each of which is of the form*

$$\pm\, \widetilde{\mathbf{B}_{i_0 \cdots i_1}}\, \widetilde{\mathbf{B}_{i_1 \cdots i_2}} \cdots \widetilde{\mathbf{B}_{i_{r-1} \cdots i_r}}, \tag{2.3}$$

where $0 = i_0 < i_1 < i_2 < \cdots < i_r = n$ and $r \le 2m + 1$.

Notice that, up to the sign, each term as expressed in Equation (2.3) is the transition matrix for a pseudorandom walk from layer 0 to layer n of the branching program, where we use $r \le m + 1$ independent draws from Nisan's generator, with the j'th draw being used to walk from layer i_{j-1} to layer i_j. In particular, the (v_0, v_{acc}) entry of Equation (2.3) equals the acceptance probability of the branching program on such a pseudorandom walk (up to the $\pm$ sign). Thus the algorithm now has the form required of a WPRG.

The seed length for the WPRG is the sum of the seed length s_{sum} needed to select a random term in the sum (using the coefficients of the WPRG to rescale the sum into an expectation) and the seed length s_{term} to generate a walk for the individual term. To select a random term in the sum requires a seed of length

$$s_{\mathrm{sum}} = \log n^{O(m)} = O(m \cdot \log(n)) = O(\log_n(1/\varepsilon) \cdot \log(n)) = O(\log(1/\varepsilon)).$$

The seed length needed for an individual term is at most

$$s_{\mathrm{term}} = O(m) \cdot s_{\mathrm{Nisan}} = O(\log_n(1/\varepsilon) \cdot \log(n) \cdot \log nw) = O(\log(1/\varepsilon) \cdot \log(n)).$$

The latter offers no improvement over Nisan's PRG. (Recall that $\varepsilon < 1/nw$.) To obtain a shorter seed length, we just need to derandomize the product in Equation (2.3). Instead of using r independent seeds, we use dependent seeds generated using the Impagliazzo–Nisan–Wigderson pseudorandom generator [27]. Specifically, we can produce a pseudorandom walk that approximates the product to within entrywise error $\pm\gamma$ using a seed of length

$$s'_{\text{term}} = s_{\text{Nisan}} + O((\log r) \cdot \log(rw/\gamma)).$$

The entrywise error of γ in each term may accumulate over the $n^{O(m)}$ terms, so to achieve a WPRG error of $O(\varepsilon)$, we should set $\gamma = \varepsilon/n^{O(m)} = 1/\varepsilon^{O(1)}$. Recalling that $r \leq 2m + 1 = O(\log_n(1/\varepsilon))$, we attain a seed length of

$$\begin{aligned} s_{\text{sum}} + s'_{\text{term}} &= O(\log(1/\varepsilon)) + O(\log n \cdot \log nw) + O(\log\log_n(1/\varepsilon) \cdot \log(1/\varepsilon)) \\ &= O(\log n \cdot \log nw + \log(1/\varepsilon) \cdot \log\log_n(1/\varepsilon)), \end{aligned}$$

which slightly improves over the bound of Braverman, Cohen, and Garg [10], and is incomparable to that of Chattopadhyay and Liao [13]. Specifically, our first term of $O(\log n \cdot \log nw)$ is better than [13] by a factor of $\log\log(nw)$, but our second term of $O(\log(1/\varepsilon) \cdot \log\log_n(1/\varepsilon))$ is worse by a factor of $\log\log_n(1/\varepsilon)$.

2.2 WPRG for permutation branching programs

Now we give an overview of our WPRG for permutation branching programs, as stated in Theorem 1.4. This is based on the the algorithm of Ahmadinejad et al. that estimates random-walk probabilities in *regular* (or even Eulerian) digraphs with better space complexity than the algorithm described in Subsection 2.1. As before, we will review their algorithm as applied to the $((n+1) \cdot w)$-vertex graph $\mathcal{G}$ associated with an ordered branching program B of length n and width w. Since we assume that the branching program B is a permutation program, the graph $\mathcal{G}$ will be 2-regular at all layers other than 0 and n. For the spectral graph-theoretic machinery used by Ahmadinejad et al., it is helpful to work with random-walk matrices that correspond to strongly connected digraphs, so we also add a complete bipartite graph of edges from layer n back to layer 0, resulting in the following modified version of the matrix $\mathbf{W}$:

$$\mathbf{W}_0 = \begin{bmatrix} 0 & \mathbf{B}_1 & 0 & \cdots & 0 \\ 0 & 0 & \mathbf{B}_2 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \ddots & \mathbf{B}_n \\ \mathbf{J}_w & 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{2.4}$$

where the $\mathbf{J}_w$ in the lower-left corner is the $w \times w$ matrix in which every entry is $1/w$ (corresponding to the complete bipartite graph we added). Notice that the matrix $\mathbf{J}_w$ is identically zero when applied to any vector that is orthogonal to the uniform distribution, so it is not very different from having 0 in the lower-left block as we had before. Indeed, the powers of $\mathbf{W}_0$ have the following form.

$$\mathbf{W}_0^2 = \begin{bmatrix} 0 & 0 & \mathbf{B}_{0..2} & 0 & 0 \\ \vdots & 0 & 0 & \ddots & 0 \\ 0 & & \vdots & & \mathbf{B}_{n-2..n} \\ \mathbf{J}_w & 0 & 0 & \cdots & 0 \\ 0 & \mathbf{J}_w & 0 & \cdots & 0 \end{bmatrix}, \ldots, \mathbf{W}_0^n = \begin{bmatrix} 0 & 0 & \cdots & 0 & \mathbf{B}_{0..n} \\ \mathbf{J}_w & 0 & & & 0 \\ 0 & \ddots & & & 0 \\ \vdots & 0 & \mathbf{J}_w & 0 & 0 \\ 0 & 0 & 0 & \mathbf{J}_w & 0 \end{bmatrix} \tag{2.5}$$

where as before

$$\mathbf{B}_{i..j} = \mathbf{B}_{i+1}\mathbf{B}_{i+2}\cdots\mathbf{B}_j.$$

Notice in particular that $\mathbf{W}_0^{n+1}$ will be a block-diagonal matrix with $\mathbf{J}_w$ on the diagonal (i. e., $\mathbf{W}_0^{n+1} = \mathbf{I}_{n+1} \otimes \mathbf{J}_w$), and thus has no dependence on the branching program B.

Now the Laplacian $\mathbf{I}_{(n+1)w} - \mathbf{W}_0$ is no longer invertible (the uniform distribution is in the kernel). In [2], they instead estimate the Moore–Penrose pseudoinverse of $\mathbf{I}_{(n+1)w} - \mathbf{W}_0$. We instead scale $\mathbf{W}_0$ by a factor $c = 1 - 1/(n+1)$, and consider the Laplacian $\mathbf{L}_0 = \mathbf{I}_{(n+1)w} - c\mathbf{W}_0$. Looking ahead, this scaling factor ensures that the condition number of $\mathbf{L}_0$ depends only on n, allowing us to obtain a seed length independent of w. Then, by the expressions above for the powers of $\mathbf{W}_0$, it can be shown that from

$$\mathbf{L}_0^{-1} = \mathbf{I}_{(n+1)w} + c\mathbf{W}_0 + c^2\mathbf{W}_0^2 + c^3\mathbf{W}_0^3 + \ldots$$

we can compute $\mathbf{B}_{0..n}$, which appears in $\mathbf{W}_0^n$ with a scaling factor $c^n \geq 1/4$.

So again to estimate the acceptance probability of B, it suffices to compute a sufficiently good approximation to $\mathbf{L}_0^{-1}$. As before, it suffices to compute a matrix $\widetilde{\mathbf{L}_0^{-1}}$ such that $\|\mathbf{I}_N - \widetilde{\mathbf{L}_0^{-1}}\mathbf{L}_0\| \leq \alpha$ for some constant $\alpha < 1$ (letting $N := (n+1)w$) and a submultiplicative matrix norm $\|\cdot\|$, because then we can use preconditioned Richardson iterations (Lemma 2.2) to estimate $\mathbf{L}_0$ to within arbitrary entrywise accuracy.

Unfortunately we don't know how to directly obtain such an initial approximation $\widetilde{\mathbf{L}_0^{-1}}$ efficiently enough for our result. Instead, following Ahmadinejad et al., we tensor $\mathbf{W}_0$ with a sufficiently long directed cycle. Specifically, we let $\mathbf{C}_i$ be the directed cycle on 2^i vertices, and consider $\mathbf{C}_q$ for $q = \log(n+1)$ (which we assume is an integer WLOG). We consider the *cycle lift*, whose transition matrix is

$$\mathbf{C}_q \otimes \mathbf{W}_0 = \begin{bmatrix} 0 & \mathbf{W}_0 & 0 & \cdots & 0 \\ 0 & 0 & \mathbf{W}_0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \ddots & \mathbf{W}_0 \\ \mathbf{W}_0 & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Then, we seek to invert the Laplacian $\mathbf{L} = \mathbf{I}_{2^q N} - c\mathbf{C}_q \otimes \mathbf{W}_0$. Similarly to the above, we have

$$\begin{aligned}
\mathbf{L}^{-1} &= \left(\mathbf{I}_{2^q N} - c\mathbf{C}_q \otimes \mathbf{W}_0\right)^{-1} \\
&= \left(\mathbf{I}_{2^q N} - c^{n+1}\mathbf{C}_q^{n+1} \otimes \mathbf{W}_0^{n+1}\right)^{-1} \cdot \left(\mathbf{I}_{2^q N} + c\mathbf{C}_q \otimes \mathbf{W}_0 + c^2\mathbf{C}_q^2 \otimes \mathbf{W}_0^2 + \cdots + c^n\mathbf{C}_q^n \otimes \mathbf{W}_0^n\right) \\
&= \left(\mathbf{I}_{2^q N} - c^{n+1}\mathbf{C}_q^{n+1} \otimes (\mathbf{I}_{n+1} \otimes \mathbf{J}_w)\right)^{-1} \cdot \left(\mathbf{I}_{2^q N} + c\mathbf{C}_q \otimes \mathbf{W}_0 + c^2\mathbf{C}_q^2 \otimes \mathbf{W}_0^2 + \cdots + c^n\mathbf{C}_q^n \otimes \mathbf{W}_0^n\right).
\end{aligned}$$

Thus, letting

$$\mathbf{M} = \mathbf{I}_{2^q N} - c^{n+1}\mathbf{C}_q^{n+1} \otimes (\mathbf{I}_{n+1} \otimes \mathbf{J}_w) = \mathbf{I}_{2^q N} - c^{n+1}\mathbf{I}_{2^q} \otimes (\mathbf{I}_{n+1} \otimes \mathbf{J}_w),$$

which has no dependence on the branching program, we obtain

$$\begin{aligned}
\mathbf{M} \cdot \mathbf{L}^{-1} &= \mathbf{I}_{2^q N} + c\mathbf{C}_q \otimes \mathbf{W}_0 + c^2\mathbf{C}_q^2 \otimes \mathbf{W}_0^2 + \cdots + c^n\mathbf{C}_q^n \otimes \mathbf{W}_0^n \\
&= \begin{bmatrix} \mathbf{I}_N & c\mathbf{W}_0 & c^2\mathbf{W}_0^2 & \cdots & c^n\mathbf{W}_0^n \\ c^n\mathbf{W}_0^n & \mathbf{I}_N & c\mathbf{W}_0 & \cdots & c^{n-1}\mathbf{W}_0^{n-1} \\ \vdots & & \ddots & & \vdots \\ c^2\mathbf{W}_0^2 & c^3\mathbf{W}_0^3 & c^4\mathbf{W}_0^4 & \ddots & c\mathbf{W}_0 \\ c\mathbf{W}_0^1 & c^2\mathbf{W}_0^2 & c^3\mathbf{W}_0^3 & \cdots & \mathbf{I}_N \end{bmatrix}.
\end{aligned}$$

Thus, if we can accurately estimate $\mathbf{L}^{-1}$, we can obtain an accurate estimate of $\mathbf{W}_0^n$, whose upper-right block equals $\mathbf{B}_{0..n}$ and thus contains the acceptance probability of the branching program.

To compute an approximate inverse of $\mathbf{L} = \mathbf{I}_{2^q N} - c\mathbf{C}_q \otimes \mathbf{W}_0$, Ahmadinejad et al. provide a recursive formula expressing $(\mathbf{I}_{2^q N} - c\mathbf{C}_q \otimes \mathbf{W}_0)^{-1}$ in terms of $(\mathbf{I}_{2^{q-1} N} - c^2\mathbf{C}_{q-1} \otimes \mathbf{W}_0^2)^{-1}$ and some applications of the matrix $\mathbf{W}_0$. That is, computing the inverse of the Laplacian of the cycle lift of $\mathbf{W}_0$ reduces to computing the inverse of the Laplacian of a cycle lift of $\mathbf{W}_0^2$ with a cycle of half the length. At the bottom of the recursion (after q levels of recursion), we need to compute the inverse of

$$\mathbf{I}_N - c^{2^q}\mathbf{W}_0^{2^q} = \mathbf{I}_N - c^{n+1}\mathbf{W}_0^{n+1} = \mathbf{I}_N - c^{n+1}\mathbf{I}_{n+1} \otimes \mathbf{J}_w,$$

which is easy (and does not depend on the branching program). The resulting formula for $(\mathbf{I}_{2^q N} - c\mathbf{C}_q \otimes \mathbf{W}_0)^{-1}$ is a polynomial in $\mathbf{W}_0, \mathbf{W}_0^2, \mathbf{W}_0^4, \ldots, \mathbf{W}_0^{2^{q-1}}$. However, computing these high powers of $\mathbf{W}_0$ exactly is too expensive in space usage.

Thus, instead Ahmadinejad et al. use the *derandomized square* [41] which allows for computing a sequence $\mathbf{W}_0, \mathbf{W}_1, \ldots, \mathbf{W}_q$ where $\mathbf{W}_i$ is a sparsification of $\mathbf{W}_{i-1}^2$ with two desirable properties. First, $\mathbf{W}_q$ can be constructed in deterministic space

$$O(\log nw + q \cdot \log(1/\delta))$$

for an error parameter δ, rather than the space $O(q \cdot \log nw)$ of exact repeated squaring. Second, they introduce a new notion of spectral approximation called *unit-circle approximation*, and show that the derandomized square $\mathbf{W}_i$ is a unit-circle approximation of $\mathbf{W}_{i-1}^2$ to within error δ. Using these repeated derandomized squares in the recursion, Ahmadinejad et al. obtain an approximate inverse $\widetilde{\mathbf{L}^{-1}}$ with the following properties.

1. The $N \times N$ blocks of $\mathbf{M} \cdot \widetilde{\mathbf{L}^{-1}}$ are each of the form $\mathbf{W}_{i_1}\mathbf{W}_{i_2}\cdots\mathbf{W}_{i_r}$ where $r = O(q)$

2. There is a submultiplicative matrix norm $\|\cdot\|_{\mathbf{F}}$ such that $\|\mathbf{I}_{2^q N} - \widetilde{\mathbf{L}^{-1}}\mathbf{L}\|_{\mathbf{F}} = O(q^2\delta)$. Moreover, achieving an $\varepsilon/\mathrm{poly}(n)$ approximation of $\mathbf{M} \cdot \mathbf{L}^{-1}$ in $\mathbf{F}$-norm implies an ε approximation of $\mathbf{M} \cdot \mathbf{L}^{-1}$ in max-norm. Ahmadinejad et al. actually lose a factor of $\mathrm{poly}(nw)$ in moving from $\mathbf{F}$-norm to approximation in max-norm, but we improve this bound to $\mathrm{poly}(n)$ by our choice of scaling factor $c = 1 - 1/(n+1)$.

Item 1 allows for constructing $\mathbf{M} \cdot \widetilde{\mathbf{L}^{-1}}$ from $\mathbf{W}_0, \mathbf{W}_1, \dots, \mathbf{W}_q$ in space

$$O(\log q \cdot \log nw).$$

By Item 2, if we take $\delta < 1/\Omega(q^2)$, we can apply preconditioned Richardson iterations (Lemma 2.2) with degree $m = O(\log(n/\varepsilon)/\log(1/q\delta))$ to obtain $\widetilde{\mathbf{L}_\varepsilon} = \mathbf{P}_m$ such that $\mathbf{M} \cdot \widetilde{\mathbf{L}_\varepsilon}$ approximates $\mathbf{M}\mathbf{L}^{-1}$ to within entrywise error ε. The preconditioned Richardson iterations have an additive space cost of

$$O(\log m \cdot \log nw).$$

Taking $\delta = 1/\Omega(q^2)$ and recalling that $q = \log(n+1)$, the final space complexity is

$$O(\log(nw) + q\log q) + O(\log q \cdot \log nw) + O(\log\log(n/\varepsilon) \cdot \log nw) = O(\log nw \cdot \log\log(n/\varepsilon)).$$

To view this algorithm as a WPRG for permutation branching programs, we use the equivalence between the Impagliazzo–Nisan–Wigderson (INW) generator on permutation branching programs and the derandomized square of the corresponding graph, as established in [41, 26]. Using this correspondence, the matrix $\mathbf{W}_i$ has the same structure as $\mathbf{W}^{2^i}$ (see Equation (2.5)), except that each block of the form $\mathbf{B}_{j..j+2^i}$ is replaced with a matrix $\widetilde{\mathbf{B}_{j..j+2^i}}$ that is the transition matrix of a pseudorandom walk from layer j of the branching program to layer $j + 2^i$ using the INW generator. The seed length to generate this pseudorandom walk is

$$s_{\mathrm{INW}} = O(q\log(q/\delta)),$$

which, as highlighted in [26], is independent of the width w of the branching program. This is the place where we use the fact that B is a permutation branching program rather than a regular branching program. Even though the algorithm of Ahmadinejad et al. works for regular directed graphs (and hence regular branching programs), the derandomized square operations used in that case can no longer be viewed as being obtained by using a pseudorandom generator to derandomize walks in the graph.

Then, again assuming without loss of generality that $\widetilde{\mathbf{B}_{(j-1)\dots j}} = \mathbf{B}_{(j-1)\dots j}$ for $j = 1, \dots, n$, we have the following analogue of Observation 2.3.

Observation 2.4. *The upper-right $w \times w$ block of $\mathbf{M} \cdot \widetilde{\mathbf{L}_\varepsilon}$ equals the sum of at most $n^{O(m)}$ terms, each of which is of the form*

$$\pm\, \widetilde{\mathbf{B}_{i_0\cdots i_1}}\widetilde{\mathbf{B}_{i_1\cdots i_2}}\cdots\widetilde{\mathbf{B}_{i_{r-1}\cdots i_r}}, \tag{2.6}$$

where $0 = i_0 < i_1 < i_2 < \cdots < i_r = n$ and $r = O(qm)$.

As in Subsection 2.1, the algorithm now has the form required of a WPRG and our only remaining challenge is to keep the seed length small. The seed length for the WPRG is the sum of the seed length needed to select a random term in the sum (using the coefficients of the WPRG to rescale the sum into an expectation) and the seed length to generate a walk for the individual term. To select a random term in the sum requires a seed of length

$$s_{\mathrm{sum}} = \log(n^{O(m)}).$$

The seed length needed for an individual term is at most

$$s_{\mathrm{term}} = O(qm) \cdot s_{\mathrm{INW}},$$

which again would be too expensive for us. To derandomize the product in Equation (2.6), we again use the INW generator, but rely on the analysis in [26] for permutation branching programs to maintain a seed length that is independent of the width. Specifically, we can produce a pseudorandom walk that approximates the product to within entrywise error $\pm\gamma$ using a seed of length

$$s'_{\mathrm{term}} = s_{\mathrm{INW}} + O((\log r) \cdot \log(\log(r)/\gamma)) = s_{\mathrm{INW}} + O(\log qm \cdot \log(\log(qm)/\gamma)).$$

The entrywise error of γ in each term may accumulate over the $n^{O(m)}$ terms, so to achieve a WPRG error of $O(\varepsilon)$, we should set $\gamma = \varepsilon/n^{O(m)}$, which means that $s'_{\mathrm{term}} \geq s_{\mathrm{sum}}$.

All in all, we attain a seed length of

$$\begin{aligned} s_{\mathrm{sum}} + s'_{\mathrm{term}} &= O(m \log n) + s_{\mathrm{INW}} + O((\log qm) \cdot \log(\log(qm)/\gamma)) \\ &= O(q \log(q/\delta)) + \widetilde{O}(m \log n) + O(\log qm \cdot \log(n/\varepsilon)) \\ &= \widetilde{O}\left(\log n \cdot \log(1/\delta) + \frac{\log(n/\varepsilon)}{\log(1/(\delta \log n))} \cdot \log n + \log\log(n/\varepsilon) \cdot \log(n/\varepsilon)\right). \end{aligned}$$

Optimizing the choice of δ as $\delta = \exp(-\widetilde{\Theta}(\sqrt{\log(n/\varepsilon)}))$, we get a seed length of

$$\widetilde{O}(\log n \sqrt{\log(n/\varepsilon)} + \log(1/\varepsilon)).$$

Note that the choice of δ here is much smaller than in the Ahmadinejad et al. algorithm, which used $\delta = 1/\mathrm{polylog}(n)$. The reason we need the smaller choice of δ is to reduce the effect of the $\log(n^{O(m)})$ price we pay in s_{sum} and s'_{term}, which does not have an analogue in the algorithm of Ahmadinejad et al.

2.3 Perspective

Some intuition for the ability of WPRGs to beat the parameters of PRGs can come from the study of *samplers* [22]. A *sampler* for a class $\mathcal{F}$ of functions $f : \{0,1\}^m \to \mathbb{R}$ is a randomized algorithm Samp that is given oracle access to a function $f \in \mathcal{F}$ and, with probability at least $1-\delta$, outputs an estimate of $\mathbb{E}[f(U_n)]$ to within additive error $\pm\varepsilon$. Most often, the class $\mathcal{F}$ is taken to be the class of all bounded functions $f : \{0,1\}^m \to [0,1]$, but some authors have considered the general definition and other classes, such as the class $\mathcal{F}$ of unbounded functions f such that the random variable $f(U_n)$ has subgaussian tails [6, 1]. Two key complexity parameters of a sampler are its *randomness complexity* (the number of coin tosses it uses, typically as a function of m, δ, and ε) and its *sample complexity* (the number of queries it makes to the oracle f). An *averaging sampler* is one that has a restricted form, where it uses its coin tosses to generate (possibly correlated) samples $x_1, \ldots, x_S$, and then outputs the average of f on the samples, i. e., $(f(x_1) + \cdots + f(x_S))/S$.

As noted by Cheng and Hoza [16], PRGs and WPRGs can be viewed as deterministic averaging samplers (i. e., with randomness complexity and failure probability zero). Specifically, a PRG $G : \{0,1\}^s \to \{0,1\}^m$ for a class $\mathcal{F}$ is a deterministic averaging sampler for the class $\mathcal{F}$ with sample complexity $S = 2^s$. Indeed, the sampler simply outputs the set of all $S = 2^s$ outputs of G. A WPRG can be viewed as a more general form of a nonadaptive deterministic sampler for the class $\mathcal{F}$, one that is restricted to output a linear combination of the function values.

So comparing the power of PRGs vs. WPRGs is a special case of the more general problem of comparing the power of averaging samplers vs. more general nonadaptive samplers. In this more general framing, there are some natural examples of classes $\mathcal{F}$ where nonadaptive samplers can have smaller sample complexity than any averaging sampler. Specifically, if we consider the class $\mathcal{F}$ of *unbounded* functions $f : \{0,1\}^m \to \mathbb{R}$ with bounded variance, i. e., $\mathrm{Var}[f(U_n)] \leq 1$, then the best sample complexity for an averaging sampler is $\min\{\widetilde{\Theta}(1/\varepsilon^2\delta), 2^{\Theta(m)}\}$. (Essentially, Chebychev's Inequality is tight for such functions.) However, there is a nonadaptive sampler with sample complexity $O(\log(1/\delta)/\varepsilon^2)$, namely the *median-of-averages sampler*, which outputs the median of $O(\log(1/\delta))$ averages, with each average being on $O(1/\varepsilon^2)$ samples (see Section 8).

This example suggests two areas of investigation. First, can we gain further benefits in seed length by considering further generalizations of PRGs that are allowed to estimate acceptance probability with more general functions than linear combinations (or possibly even with adaptive queries)? Some examples are the line of work on converting hitting-set generators for circuits [3, 4, 12, 23] or ordered branching programs [16] into deterministic samplers. Second, is there a benefit in the study of samplers in restricting attention to ones that output linear combinations like WPRGs? Perhaps these still retain some of the useful composition properties and connections to other pseudorandom objects that are enjoyed by averaging samplers (see [47, 45, 1]), while allowing for gains in sample and/or randomness complexity.

2.4 Organization of the remaining sections

In Section 3 we introduce arithmetic over WPRGs and the view of branching programs as matrix-valued functions. In Section 4 we prove Theorem 4.1, using a simple analysis that introduces preconditioning methods. In Section 5 we prove Theorem 1.4, using more sophisticated preconditioning tools from [2] and [19] and the analysis of the INW PRG from [26].

3 Preliminaries

Following [39], we will view branching programs not as boolean functions, but as matrix-valued functions $\mathbf{B} : \{0,1\}^n \to \mathbb{R}^{w\times w}$ where $\mathbf{B}[s]_{i,j} = 1$ if the branching program started at state i ends at state j upon reading input s. In all cases, we will index the rows and columns of matrices starting from *zero*.

Definition 3.1. Let B be a width-w, length-n branching program with transition functions $B_1, \ldots, B_n$. For $t \in [n]$ let $\mathbf{B}_t : \{0,1\} \to \mathbb{R}^{w\times w}$ be defined as

$$\mathbf{B}_t[s]_{a,b} = \begin{cases} 1 & \text{if } B_t(a,s) = b \\ 0 & \text{otherwise} \end{cases}$$

For $0 \le i < j \le n$ let $\mathbf{B}_{i..j}$ be defined via matrix multiplication as

$$\mathbf{B}_{i..j}[s_{i+1} \ldots s_j] = \mathbf{B}_{i+1}[s_{i+1}] \cdots \mathbf{B}_j[s_j]$$

and let $\mathbf{B} = \mathbf{B}_{0..n}$. Observe that $\mathbf{B}_{i..j}[s_{i+1} \ldots s_j]_{u,v} = 1$ if and only if $\mathbf{B}$ reaches state v in layer j when started in state u in layer i and reading $(s_{i+1} \ldots s_j)$. Define the constant function on zero bits $\mathbf{B}_{i,i}[] = \mathbf{I}_w$ for all i. This is purely for cleanliness of later derivations.

We now formally define weighted distributions.

Definition 3.2. A **weighted distribution** (a. k. a. **pseudodistribution**) on $\{0,1\}^n$ is a jointly distributed random variable (X,Y) taking values in $\{0,1\}^n \times \mathbb{R}$. For a (possibly matrix-valued) function $f : \{0,1\}^n \to \mathbb{R}^d$, $f[(X,Y)]$ denotes the random variable $Y \cdot f(X)$ and $\overline{f}[(X,Y)]$ its expectation. For a weighted generator $(G,\rho) : \{0,1\}^s \to \{0,1\}^n \times \mathbb{R}$ we write $\overline{f}[(G,\rho)]$ as shorthand for $\mathbb{E}_{x\leftarrow U_s}[\rho(x) \cdot f(G(x))]$. We say a weighted generator (G,ρ) is r**-bounded** if $|\rho| \le r$.

We can now define matrix-valued functions evaluated on distributions or weighted distributions. We write $U_n = U_{\{0,1\}^n}$ for convenience.

Definition 3.3. Let (X,Y) be a weighted distribution on $\{0,1\}^n$ and $\|\cdot\|$ a norm on $w \times w$ matrices and let $\mathcal{B}$ be a class of length-n, width-w ordered branching programs. We say that (X,Y) is ε**-pseudorandom** for $\mathcal{B}$ with respect to $\|\cdot\|$ if for all $\mathbf{B} \in \mathcal{B}$ we have

$$\left\| \overline{\mathbf{B}}[(X,Y)] - \overline{\mathbf{B}}[U_n] \right\| \le \varepsilon.$$

A weighted PRG generates a weighted distribution, exactly analogous to a PRG generating a distribution.

Definition 3.4. A weighted generator $(G,\rho) : \{0,1\}^s \to \{0,1\}^n \times \mathbb{R}$ is ε**-pseudorandom** for $\mathcal{B}$ with respect to $\|\cdot\|$ if the weighted distribution $(G(U_s), \rho(U_s))$ is ε-pseudorandom for $\mathcal{B}$ with respect to $\|\cdot\|$. That is, for all $\mathbf{B} \in \mathcal{B}$, $\|\overline{\mathbf{B}}[(G,\rho)] - \overline{\mathbf{B}}[U_n]\| \le \varepsilon$. We also say (G,ρ) is an ε-**weighted pseudorandom generator** (ε-**WPRG**) for $\mathcal{B}$ with respect to $\|\cdot\|$.

To use this definition, we need to select a matrix norm. We will work with several different norms on matrices $\mathbf{A} \in \mathbb{R}^{w \times w}$. Some examples include

- $\|\mathbf{A}\|_{\max} = \max_{i,j} |\mathbf{A}_{i,j}|$
- $\|\mathbf{A}\|_1 = \max_{x \in \mathbb{R}^w \setminus \{0\}} \|x\mathbf{A}\|_1 / \|x\|_1 = \max_i \|\mathbf{A}_{i,\cdot}\|$ where $\mathbf{A}_{i,\cdot}$ is the ith row of $\mathbf{A}$. We note that we define the norm in terms of left multiplication.
- $\|\mathbf{A}\|_2 = \max_{x \in \mathbb{R}^w \setminus \{0\}} \|x\mathbf{A}\|_2 / \|x\|_2 = \sigma_{\max}(\mathbf{A})$ where $\sigma_{\max}(\mathbf{A})$ is the maximum singular value of $\mathbf{A}$.

Above and throughout the paper, all vectors are *row* vectors.

Lemma 3.5. *Suppose (X, Y) is ε-pseudorandom for a class $\mathcal{B}$ of branching programs with respect to $\|\cdot\|_{\max}$. Then*

1. *(X, Y) is ε-pseudorandom for the class of boolean functions (according to Definition 1.1) obtained by selecting $B \in \mathcal{B}$, choosing any start vertex $v_0 \in [w]$ and a single accept vertex $v_{acc} \in [w]$.*
2. *(X, Y) is $w \cdot \varepsilon$-pseudorandom for the class of boolean functions (again according to Definition 1.1) obtained by selecting $B \in \mathcal{B}$, choosing any start vertex $v_0 \in [w]$ and a set $V_{acc} \subseteq [w]$ of accept vertices.*

Proof. Unwinding the definitions we have $\| \mathbb{E}[Y \cdot \mathbf{B}[X]] - \mathbb{E}[\mathbf{B}[U_n]]\|_{\max} \le \varepsilon$, which implies

$$|\mathbb{E}[Y \cdot \mathbf{B}[X]_{v_0, v_{\mathrm{acc}}}] - \mathbb{E}[\mathbf{B}[U_n]_{v_0, v_{\mathrm{acc}}}]| \le \varepsilon$$

for all states v_0, v_{acc}. Since $\mathbf{B}[s]_{v_0, v_{\mathrm{acc}}} : \{0,1\}^n \to \{0,1\}$ is precisely the boolean function obtained by choosing start vertex v_0 and accept vertex v_{acc} for B, this shows the first bound. For the second case, enumerating over all states $v \in V_{\mathrm{acc}}$ and applying a union bound completes the proof. □

Thus, our end goal is to construct WPRGs with respect to $\|\cdot\|_{\max}$. But at intermediate stages in our analysis we work with other norms, such as the ℓ_2 and ℓ_1 norms, because they are submultiplicative (i. e., $\|AB\| \le \|A\| \cdot \|B\|$) and max-norm is not.

We can define rules for “arithmetic” on WPRGs, which naturally translate into operations on matrix forms. Below and throughout the paper, all logs are base 2.

Definition 3.6 (Sum Rule for WPRGs)**.** Given WPRGs $F_a = (G_a, \rho_a), F_b = (G_b, \rho_b)$ each with seed length s, let $F_a + F_b$ be the WPRG with seed length $s+1$, where for $(x, y) \in \{0,1\}^s \times \{0,1\}$ we define

$$(F_a + F_b)((x, y)) = \begin{cases} (G_a(x), 2\rho_a(x)) & y = 1 \\ (G_b(x), 2\rho_b(x)) & y = 0 \end{cases}$$

Lemma 3.7. *For WPRGs F_a, F_b as defined above, for every branching program* $\mathbf{B}$

$$\overline{\mathbf{B}}[F_a] + \overline{\mathbf{B}}[F_b] = \overline{\mathbf{B}}[F_a + F_b].$$

Furthermore, if F_a and F_b are explicit then $F_a + F_b$ is, and if F_a and F_b are r-bounded then $F_a + F_b$ is $2r$-bounded.

Proof. The explicitness and boundedness properties are immediate from the definition. Then

$$\begin{aligned}\overline{\mathbf{B}}[F_a] + \overline{\mathbf{B}}[F_b] &= \mathop{\mathbb{E}}_{x \leftarrow U_s} \rho_a(x)\mathbf{B}[G_a(x)] + \mathop{\mathbb{E}}_{x \leftarrow U_s} \rho_b(x)\mathbf{B}[G_b(x)] \\ &= \frac{1}{2^{s+1}} \sum_{x \in \{0,1\}^s} (2\rho_a(x)\mathbf{B}[G_a(x)] + 2\rho_b(x)\mathbf{B}[G_b(x)]) \\ &= \overline{\mathbf{B}}[F_a + F_b]\,. \end{aligned}$$

□

We frequently take sums over large collections of WPRGs, so we state a recursive definition of the addition rule.

Proposition 3.8. *Let $\{F_i : \{0,1\}^s \to \{0,1\}^n : i \in [V]\}$ be a set of (explicit) WPRGs where given i, F_i can be evaluated in space $O(s)$. Then $\sum_{i=1}^{V} F_i$ is an explicit $2V$−bounded WPRG with seed length $s + \lceil \log(V) \rceil$.*

Proof. We can recursively construct the WPRGs $F_a = \sum_{i=1}^{\lfloor V/2 \rfloor} F_i$ and $F_b = \sum_{i=\lfloor V/2 \rfloor+1}^{V} F_i$, which by induction have seed length at most $s + \lceil \log(V/2) \rceil$, and apply Definition 3.6 (padding the seed length of F_b to make it equal to the seed length of F_a if necessary) to obtain seed length $s + \lceil \log(V/2) \rceil + 1 = s + \lceil \log(V) \rceil$. □

Definition 3.9 (Product Rule for WPRGs)**.** Given WPRGs $F_a = (G_a, \rho_a)$, $F_b = (G_b, \rho_b)$ each with seed length s and output lengths n, n', respectively, let $F_a F_b$ be the WPRG with seed length $2s$ and output length $n + n'$, where for $(x, y) \in \{0,1\}^s \times \{0,1\}^s$ we define

$$(F_a F_b)((x, y)) = (G_a(x) \| G_b(y), \rho_a(x)\rho_b(y))$$

where $\|$ denotes concatenation. We define the product of a WPRG F_a and scalar $\lambda \in \mathbb{R}$ as $(\lambda F_a)(x) = (G_a(x), \lambda \cdot \rho_a(x))$.

Lemma 3.10. *For WPRGs F_a, F_b as defined above, for every pair of branching programs $\mathbf{B}, \mathbf{B}'$ of lengths n, n' and equal width w,*

$$(\overline{\mathbf{B}\mathbf{B}'})[F_a F_b] = \overline{\mathbf{B}}[F_a]\overline{\mathbf{B}}'[F_b].$$

Furthermore, if F_a and F_b are explicit then $F_a F_b$ is, and if F_a and F_b are r-bounded then $F_a F_b$ is r^2-bounded.

Proof. The explicitness and boundedness properties are immediate from the definition. Then

$$\begin{aligned}\overline{\mathbf{B}}[F_a]\overline{\mathbf{B}}'[F_b] &= \mathop{\mathbb{E}}_{x \leftarrow U_s} \rho_a(x)\mathbf{B}[G_a(x)] \mathop{\mathbb{E}}_{y \leftarrow U_s} \rho_b(y)\mathbf{B}'[G_b(y)] \\ &= \mathop{\mathbb{E}}_{x,y \leftarrow U_s} \rho_a(x)\rho_b(y)\mathbf{B}[G_a(x)]\mathbf{B}'[G_b(y)] \\ &= (\overline{\mathbf{B}\mathbf{B}'})[F_a F_b] \end{aligned}$$

□

We implicitly define the sum and product of WPRGs with different seed lengths, by first padding the shorter seed to equal that of the longer.

4 Pseudodistributions for general branching programs

We first develop the WPRG for general ordered branching programs, as it outlines some of the ideas and constructions used in our main result (Theorem 1.4) but with simpler analysis.

Theorem 4.1. *For all $n, w \in \mathbb{N}$ and $\varepsilon \in (0, 1/2)$, there exists an explicit ε-WPRG for the class of ordered branching programs of length n and width w with respect to $\|\cdot\|_{\max}$ with seed length*

$$s = O(\log(n)\log(nw) + \log(1/\varepsilon)\log\log_n(1/\varepsilon)).$$

Moreover, the generator is $\mathrm{poly}(1/\varepsilon)$ *bounded.*

4.1 A WPRG with large seed length

In this subsection, we construct an explicit WPRG for ordered branching programs with large seed length.

Lemma 4.2. *Given $n, w \in \mathbb{N}$ and $\varepsilon \in (0, 1/2)$, define $\ell = \lceil \log_n(1/\varepsilon) \rceil + 1$. Then there exists an explicit weighted generator GEN_0 such that GEN_0 is ε-pseudorandom for the class of ordered branching programs of length n and width w with respect to $\|\cdot\|_{\max}$ and*

$$\mathrm{GEN}_0 = \sum_{i \in [V]} \tau_i \cdot P_{i,1} P_{i,2} \cdots P_{i,k}$$

such that

1. $V = n^{O(\ell)} = \mathrm{poly}(1/\varepsilon)$
2. $k = O(\ell)$
3. *For all i, $\tau_i \in \{-1, 1\}$.*
4. *For all i, j, $P_{i,j}$ is an (unweighted) PRG with seed length $s = O(\log n \cdot \log(nw))$.*
5. *Given $i \in [V]$ and $j \in [k]$, τ_i and $P_{i,j}$ are evaluatable in space $O(s + \log V)$.*

Applying the sum and product rules to the output of the lemma, we obtain an ε-WPRG for ordered branching programs with seed length $O(\ell \cdot \log n \cdot \log(nw) + \log V) = O(\log(nw/\varepsilon)\log n)$, no better than the Nisan PRG. However, since $k = O(\log_n(1/\varepsilon))$ we will later apply standard derandomization results to shrink the seed length of each summand.

The pseudorandom generators $P_{i,j}$ in Lemma 4.2 are instantiations of Nisan's classic generator.

Theorem 4.3 ([34])**.** *For all n, w and $\varepsilon \in (0, 1/2)$, there exists an explicit ε-PRG for the class of ordered branching programs of length n and width w with respect to $\|\cdot\|_1$ with seed length $s = O(\log n \cdot \log(nw/\varepsilon))$.*

We will use Nisan's generator to approximate the random walk matrix of an arbitrary branching program B.

Definition 4.4. Let R be the trivial PRG on one bit. Given $n, w \in \mathbb{N}$, for every length-n, width-w branching program $\mathbf{B}$, define the **random walk matrix W** of $\mathbf{B}$ as the $(n+1) \times (n+1)$ block matrix

$$\mathbf{W} = \begin{bmatrix} 0 & \overline{\mathbf{B}}_{0..1}[R] & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \overline{\mathbf{B}}_{n-1..n}[R] \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

And the **Laplacian L** of $\mathbf{B}$ as

$$\mathbf{L} = \mathbf{I}_{(n+1)w} - \mathbf{W} = \begin{bmatrix} \mathbf{I}_w & -\overline{\mathbf{B}}_{0..1}[R] & 0 & 0 \\ 0 & \mathbf{I}_w & \ddots & 0 \\ 0 & 0 & \mathbf{I}_w & -\overline{\mathbf{B}}_{n-1..n}[R] \\ 0 & 0 & 0 & \mathbf{I}_w \end{bmatrix}.$$

The inverse of the Laplacian of B holds information about random walk probabilities, which we will exploit.

Remark 4.5. Let $\mathbf{L}$ be the Laplacian of a length-n, width-w branching program $\mathbf{B}$. Then $\mathbf{L}^{-1}$ has the form

$$\mathbf{L}^{-1}_{i,j} = \begin{cases} \overline{\mathbf{B}}_{i..j}[U_{j-i}] & i < j \\ \mathbf{I}_w & i = j \\ 0 & i > j \end{cases}.$$

Note in particular that the $(0, n)$ block of $\mathbf{L}^{-1}$ equals $\overline{\mathbf{B}}[U_n]$, the distribution of truly random input over the branching program. To obtain a high-quality estimate of $\mathbf{L}^{-1}$, we first obtain a weak estimate by substituting truly random input for the output of Nisan's PRG.

Proposition 4.6. *Let* $\mathbf{L}$ *be the Laplacian of a length-n, width-w branching program* $\mathbf{B}$*. Then for every* $k \in [n]$*, let* NIS_k *be the PRG obtained from Theorem 4.3 with width w, length r and error* $\delta = 1/4n^2$*. Define*

$$\widetilde{\mathbf{L}^{-1}}_{i,j} = \begin{cases} \overline{\mathbf{B}}_{i..j}[\mathrm{NIS}_{j-i}] & i < j \\ \mathbf{I}_w & i = j \\ 0 & i > j \end{cases}.$$

Let $\mathbf{Err} = \mathbf{I}_{(n+1)w} - \widetilde{\mathbf{L}^{-1}}\mathbf{L}$*. Then* $\|\mathbf{Err}\|_1 \le 1/2n$*.*

Note that the set NIS_r of PRGs has no dependence on the branching program $\mathbf{B}$. To prove that the error matrix has small norm, we describe its structure.

Lemma 4.7. *Given* $n, w \in \mathbb{N}$*, let* $\mathbf{L}$ *be the Laplacian of a length-n, width-w branching program* $\mathbf{B}$*, and let* $\widetilde{\mathbf{L}^{-1}}$ *and* $\mathbf{Err}$ *be as defined in Proposition 4.6. Then viewing* $\mathbf{Err} \in \mathbb{R}^{(n+1)w \times (n+1)w}$ *as an* $(n+1) \times (n+1)$ *block matrix, we have*

$$\mathbf{Err}_{i,j} = \begin{cases} \overline{\mathbf{B}}_{i..j}[\mathrm{NIS}_{j-i-1}\, R - \mathrm{NIS}_{j-i}] & i < j \\ 0 & i \ge j \end{cases}.$$

Proof. We first consider when $i < j$.

$$\begin{aligned}
\mathbf{Err}_{i,j} &= -(\widetilde{\mathbf{L}^{-1}}\mathbf{L})_{i,j} \\
&= -\sum_{k=0}^{n} \widetilde{\mathbf{L}^{-1}}_{i,k}\mathbf{L}_{k,j} \\
&= -\left(\widetilde{\mathbf{L}^{-1}}_{i,j} \cdot \mathbf{L}_{j,j} + \widetilde{\mathbf{L}^{-1}}_{i,j-1} \cdot \mathbf{L}_{j-1,j}\right) \\
&= -\overline{\mathbf{B}}_{i..j}[\mathrm{NIS}_{j-i}] \cdot \mathbf{I}_w + \overline{\mathbf{B}}_{i..j-1}[\mathrm{NIS}_{j-i-1}]\overline{\mathbf{B}}_{j-1..j}[R] \\
&= \overline{\mathbf{B}}_{i..j}[\mathrm{NIS}_{j-i-1}\, R - \mathrm{NIS}_{j-i}]\,.
\end{aligned}$$

And for $i = j$ we have

$$\begin{aligned}
\mathbf{Err}_{i,i} &= \mathbf{I}_{(n+1)w} - (\widetilde{\mathbf{L}^{-1}}\mathbf{L})_{i,i} \\
&= \mathbf{I}_{(n+1)w} - \sum_{k=0}^{n} \widetilde{\mathbf{L}^{-1}}_{i,k}\mathbf{L}_{k,i} \\
&= \mathbf{I}_w - \widetilde{\mathbf{L}^{-1}}_{i,i}\mathbf{L}_{i,i} \\
&= 0\,.
\end{aligned}$$

The $i > j$ case is immediate, so **Err** has the desired form. □

We can then prove Proposition 4.6.

Proof. We bound the 1 norm of each block of **Err** and then take a union bound over the at most n nonzero blocks in each row. Diagonal and lower triangular blocks of **Err** are identically zero from Lemma 4.7. For every $i < j$ we have

$$\begin{aligned}
\|\mathbf{Err}_{i,j}\|_1 &= \|\overline{\mathbf{B}}_{i..j}[\mathrm{NIS}_{j-i-1}\, R - \mathrm{NIS}_{j-i}]\|_1 && \text{(Lemma 4.7)} \\
&\le \|\overline{\mathbf{B}}_{i..j}[\mathrm{NIS}_{j-i-1}\, R] - \overline{\mathbf{B}}_{i..j}[U_{j-i}]\|_1 + \|\overline{\mathbf{B}}_{i..j}[U_{j-i}] - \overline{\mathbf{B}}_{i..j}[\mathrm{NIS}_{j-i}]\|_1 \\
&\le \|\overline{\mathbf{B}}_{i..j-1}[\mathrm{NIS}_{j-i-1}] - \overline{\mathbf{B}}_{i..j-1}[U_{j-i-1}]\|_1 \cdot \|\overline{\mathbf{B}}_{j-1..j}[U_1]\|_1 \\
&\qquad + \|\overline{\mathbf{B}}_{i..j}[U_{j-i}] - \overline{\mathbf{B}}_{i..j}[\mathrm{NIS}_{j-i}]\|_1 && \text{(Submultiplicativity)} \\
&\le \frac{1}{4n^2} + \frac{1}{4n^2} = \frac{1}{2n^2}\,. && \text{(Theorem 4.3)}
\end{aligned}$$

where the third inequality uses that $\overline{\mathbf{B}}_{j-1..j}[U_1]$ is row-stochastic hence $\|\overline{\mathbf{B}}_{j-1..j}[U_1]\| \le 1$. Thus $\|\mathbf{Err}\|_1 \le n \cdot (1/2n^2) = 1/2n$ as desired. □

Therefore, by replacing truly random input with a PRG of the correct length we obtain a weak approximation of $\mathbf{L}^{-1}$. Following [2] we use preconditioned Richardson iteration to boost this to a high-quality approximation, and by describing this output in terms of a WPRG prove Lemma 4.2.

Lemma 4.8 (preconditioned Richardson iteration, [2] Lemma 6.2). *Let $\|\cdot\|$ be a submultiplicative norm on $N \times N$ real matrices. Given matrices $\mathbf{A}, \mathbf{P}_0 \in \mathbb{R}^{N\times N}$ such that $\|\mathbf{I}_N - \mathbf{P}_0\mathbf{A}\| \le \alpha$ for some $\alpha > 0$, let $\mathbf{P}_m = \sum_{i=0}^{m}(\mathbf{I}_N - \mathbf{P}_0\mathbf{A})^i\mathbf{P}_0$. Then $\|\mathbf{I}_N - \mathbf{P}_m\mathbf{A}\| \le \alpha^{m+1}$.*

Proof. We have $\mathbf{I}_N - \mathbf{P}_m\mathbf{A} = (\mathbf{I}_N - \mathbf{P}_0\mathbf{A})^{m+1}$, and then the proof follows by the submultiplicativity of $\|\cdot\|$. □

We now apply this to boost our weak estimate of $\mathbf{L}^{-1}$ to a strong estimate.

Lemma 4.9. *For every $n, w \in \mathbb{N}$ and $\varepsilon \in (0, 1/2)$, set $\ell = \lceil \log_n(1/\varepsilon)\rceil + 1$. Then for every length-$n$, width-$w$ ordered branching program $\mathbf{B}$ with Laplacian $\mathbf{L}$, let $\widetilde{\mathbf{L}^{-1}}$ and $\mathbf{Err}$ be defined as in Proposition 4.6. Then*

$$\left\| \sum_{i=0}^{\ell} \mathbf{Err}^i \cdot \widetilde{\mathbf{L}^{-1}} - \mathbf{L}^{-1} \right\|_{\max} \le \varepsilon/2.$$

Proof. We apply Lemma 2.2 with $\mathbf{A} = \mathbf{L}$, $\mathbf{P}_0 = \widetilde{\mathbf{L}^{-1}}$, $\|\cdot\| = \|\cdot\|_1$ and $\alpha \le 1/2n$ (which follows from Proposition 4.6) and obtain $\mathbf{P}_m = \sum_{i=0}^{\ell} \mathbf{Err}^i \cdot \widetilde{\mathbf{L}^{-1}}$ satisfying $\|\mathbf{I} - \mathbf{P}_m\mathbf{L}\|_1 \le \varepsilon/2n$. Finally,

$$\begin{aligned}
\|\mathbf{P}_m - \mathbf{L}^{-1}\|_{\max} &\le \|\mathbf{P}_m - \mathbf{L}^{-1}\|_1 \\
&= \|(\mathbf{I} - \mathbf{P}_m\mathbf{L})\mathbf{L}^{-1}\|_1 \\
&\le \|\mathbf{I} - \mathbf{P}_m\mathbf{L}\|_1 \cdot \|\mathbf{L}^{-1}\|_1 \\
&\le \frac{\varepsilon}{2n}(n+1)\,.
\end{aligned}$$
□

Given this error guarantee, it remains to interpret the "output" of Richardson iteration in an oblivious manner. Intuitively, taking powers of the $\mathbf{Err}$ matrix corresponds to concatenating WPRGs to create more complex WPRGs on layers. We first define an index set for products of combinations of WPRGs. The index set is equivalent to all possible divisions of the layers $\{0, \ldots, t\}$ for all $t \le n$ into at most ℓ sections.

Definition 4.10. Given $n, \ell \in \mathbb{N}$, define the index set $\mathbb{V}_{n,\ell}$ as

$$\mathbb{V}_{n,\ell} = \{(d_1, \ldots, d_r) : d_i \in \mathbb{Z}^+, \quad 0 \le r \le \ell, \quad \sum_{i=1}^{r} d_i \le n\}.$$

For $\sigma = (d_1, \ldots, d_r) \in \mathbb{V}_{n,\ell}$ we write $|\sigma| = r$. Note that this includes the empty tuple where $r = 0$.

Then the nonzero summands in the output of preconditioned Richardson iteration correspond to WPRGs indexed by $\mathbb{V}_{n,\ell}$.

Lemma 4.11. *For all $n, w, \ell \in \mathbb{N}$, let $\mathbb{V}_{n,\ell}$ be defined as in Definition 4.10 with the same n and ℓ and for all $k \in [n]$ let NIS_k be defined as in Proposition 4.6 with the same n, w. For all $\sigma = (d_1, \ldots, d_r) \in \mathbb{V}_{n,\ell}$, let $t = \sum_{i=1}^{r} d_i$ and define the WPRG (using the sum and product rules of Definitions 3.6 and 3.9)*

$$M_\sigma = \prod_{i=1}^{r} \left(\mathrm{NIS}_{d_i - 1}\, R - \mathrm{NIS}_{d_i}\right) \mathrm{NIS}_{n-t}\,.$$

Then for every length-n, width-w branching program $\mathbf{B}$ *with Laplacian* $\mathbf{L}$*, let* $\mathbf{Err}$ *and* $\widetilde{\mathbf{L}^{-1}}$ *be defined as in Proposition 4.6. Then we have*

$$\left(\sum_{r=0}^{\ell} \mathbf{Err}^r \cdot \widetilde{\mathbf{L}^{-1}}\right)_{0,n} = \sum_{\sigma \in \mathbb{V}_{n,\ell}} \overline{\mathbf{B}}[M_\sigma].$$

Proof. Fix any $1 \le r \le \ell$. Then we have

$$\begin{aligned}
&(\mathbf{Err}^r \cdot \widetilde{\mathbf{L}^{-1}})_{0,n} \\
&\quad = \sum_{t_i \in \{0..n\}^r} \mathbf{Err}_{0,t_1} \left(\prod_{i=1}^{r-1} \mathbf{Err}_{t_i,t_{i+1}}\right) \widetilde{\mathbf{L}^{-1}}_{t_r,n} \\
&\quad = \sum_{0=t_0<\ldots<t_r\le n} \left(\prod_{i=0}^{r-1} \mathbf{Err}_{t_i,t_{i+1}}\right) \widetilde{\mathbf{L}^{-1}}_{t_r,n} && \text{(Lemma 4.7)} \\
&\quad = \sum_{0=t_0<\ldots<t_r\le n} \left(\prod_{i=0}^{r-1} \overline{\mathbf{B}}_{t_i..t_{i+1}}[\mathrm{NIS}_{t_{i+1}-t_i-1}\, R - \mathrm{NIS}_{t_{i+1}-t_i}]\right) \overline{\mathbf{B}}_{t_r..n}[\mathrm{NIS}_{n-t_r}] && \text{(Lemma 4.7)} \\
&\quad = \sum_{0=t_0<\ldots<t_r\le n} \overline{\mathbf{B}}\left[\left(\prod_{i=0}^{r-1} \mathrm{NIS}_{t_{i+1}-t_i-1}\, R - \mathrm{NIS}_{t_{i+1}-t_i}\right) \mathrm{NIS}_{n-t_r}\right] && \text{(Definition 3.9)} \\
&\quad = \sum_{(d_i)_{i\in[r]}: t=\sum_{i=1}^r d_i \le n} \overline{\mathbf{B}}\left[\left(\prod_{i=1}^{r} \mathrm{NIS}_{d_i-1}\, R - \mathrm{NIS}_{d_i}\right) \mathrm{NIS}_{n-t}\right] \\
&\quad = \sum_{\sigma \in \mathbb{V}_{n,\ell}, |\sigma|=r} \overline{\mathbf{B}}[M_\sigma]\,.
\end{aligned}$$

For $r = 0$ we have

$$(\mathbf{Err}^0 \cdot \widetilde{\mathbf{L}^{-1}})_{0,n} = \widetilde{\mathbf{L}^{-1}}_{0,n} = \overline{\mathbf{B}}[\mathrm{NIS}_n] = \overline{\mathbf{B}}[M_{()}].$$

Thus,

$$\left(\sum_{r=0}^{\ell} \mathbf{Err}^r \cdot \widetilde{\mathbf{L}^{-1}}\right)_{0,n} = \sum_{r=0}^{\ell} \sum_{\sigma \in \mathbb{V}_{n,\ell}, |\sigma|=r} \overline{\mathbf{B}}[M_\sigma] = \sum_{\sigma \in \mathbb{V}_{n,\ell}} \overline{\mathbf{B}}[M_\sigma]. \qquad \square$$

Furthermore, this family of WPRGs is of the form required for Lemma 4.2.

Corollary 4.12. *Given* $n, w, \ell \in \mathbb{N}$*, let* $\mathbb{V}_{n,\ell}$ *be defined as in Definition 4.10 with the same* n *and* ℓ *and let* $\{M_\sigma : \sigma \in \mathbb{V}_{n,\ell}\}$ *be defined as in Lemma 4.11 with the same* n, w, ℓ*. For all* $\sigma = (d_1, \ldots, d_r) \in \mathbb{V}_{n,\ell}$ *we have*

$$M_\sigma = \sum_{x \in \{0,1\}^r} \tau_{\sigma,x} \cdot P_{\sigma,x,1} \cdots P_{\sigma,x,r+1},$$

where for all σ, x, i, $\tau_{\sigma,x} \in \{-1, 1\}$ *and* $P_{\sigma,x,i}$ *is an explicit PRG with seed length* $s = O(\log n \cdot \log(nw))$*. Furthermore given* $\sigma = (d_1, \ldots, d_r) \in \mathbb{V}_{n,\ell}$, $x \in \{0,1\}^r$ *and* $i \in [r+1]$, $\tau_{\sigma,x}$ *can be computed and* $P_{\sigma,x,i}$ *can be evaluated in space* $O(s + \log(|\mathbb{V}_{n,\ell}| \cdot r))$.

Proof. For all $\sigma \in \mathbb{V}_{n,\ell}$ and $x \in \{0,1\}^r$, let $\tau_{\sigma,x} = (-1)^{\sum_{i=1}^r x_i}$. For all $i \in [r]$, define

$$P_{\sigma,x,i} = \begin{cases} \mathrm{NIS}_{d_i-1} R & x_i = 0 \\ \mathrm{NIS}_{d_i} & x_i = 1 \end{cases}$$

and letting $l = \sum_{i=1}^r d_i$, define $P_{\sigma,x,r+1} = \mathrm{NIS}_{n-l}$. Then by construction

$$M_\sigma = \sum_{x \in \{0,1\}^r} \tau_{\sigma,x} \cdot P_{\sigma,x,1} \cdots P_{\sigma,x,r+1}.$$

Given any σ, x, i we have from Theorem 4.3 and Definition 3.9 that $P_{\sigma,x,i}$ is an explicit PRG with the desired seed length. ☐

We can now prove the main lemma of this subsection.

Lemma 4.13. *Given $n, w \in \mathbb{N}$ and $\varepsilon \in (0, 1/2)$, define $\ell = \lceil \log_n(1/\varepsilon) \rceil + 1$. Then there exists an explicit weighted generator GEN_0 such that GEN_0 is ε-pseudorandom for the class of ordered branching programs of length n and width w with respect to $\|\cdot\|_{\max}$ and*

$$\mathrm{GEN}_0 = \sum_{i \in [V]} \tau_i \cdot P_{i,1} P_{i,2} \cdots P_{i,k}$$

such that

1. $V = n^{O(\ell)} = \mathrm{poly}(1/\varepsilon)$
2. $k = O(\ell)$
3. *For all i, $\tau_i \in \{-1, 1\}$.*
4. *For all i, j, $P_{i,j}$ is an (unweighted) PRG with seed length $s = O(\log n \cdot \log(nw))$.*
5. *Given $i \in [V]$ and $j \in [k]$, τ_i and $P_{i,j}$ are evaluatable in space $O(s + \log V)$.*

Proof. Note that we can assume $\varepsilon = 1/n^{\Omega(1)}$ since otherwise the statement is satisfied by a single Nisan PRG. Let $\{M_\sigma : \sigma \in \mathbb{V}_{n,\ell}\}$ be defined as in Lemma 4.11 with the same n, ℓ, and let

$$\{\tau_{\sigma,x} \cdot P_{\sigma,x,1} \cdots P_{\sigma,x,r+1} : \sigma \in \mathbb{V}_{n,\ell}, x \in \{0,1\}^{|\sigma|}\}$$

be the family obtained from Corollary 4.12 ranging over σ. Then let $[V]$ be the set of terms (σ, x), let $k = r + 1 = O(\ell)$, and define

$$\mathrm{GEN}_0 = \sum_{i \in [V]} \tau_i \cdot P_{i,1} \cdots P_{i,k}\,.$$

All explicitness and seed length conditions are satisfied from Corollary 4.12, and we have $V = n^{O(\ell)} = \mathrm{poly}(1/\varepsilon)$ and GEN is $2V = \mathrm{poly}(1/\varepsilon)$ bounded as desired. Now fix an arbitrary length-n, width-w branching program $\mathbf{B}$ with Laplacian $\mathbf{L}$ and let $\widetilde{\mathbf{L}^{-1}}$ and $\mathbf{Err}$ be as defined as in Proposition 4.6. Then

$$\begin{aligned}
\varepsilon/2 &\geq \left\| \left(\sum_{i=0}^{\ell} \mathbf{Err}^i \cdot \widetilde{\mathbf{L}^{-1}} \right)_{0,n} - \left(\mathbf{L}^{-1} \right)_{0,n} \right\|_{\max} && \text{(Lemma 4.9)} \\
&= \left\| \sum_{\sigma \in \mathbb{V}_{n,\ell}} \overline{\mathbf{B}}[M_\sigma] - \left(\mathbf{L}^{-1} \right)_{0,n} \right\|_{\max} && \text{(Lemma 4.11)} \\
&= \left\| \overline{\mathbf{B}} \left[\sum_{\sigma \in \mathbb{V}_{n,\ell}} M_\sigma \right] - \overline{\mathbf{B}}[U_n] \right\|_{\max} && \text{(Remark 4.5)} \\
&= \left\| \overline{\mathbf{B}} \left[\sum_{i \in [V]} \tau_i \cdot P_{i,1} \cdots P_{i,k} \right] - \overline{\mathbf{B}}[U_n] \right\|_{\max} && \text{(Corollary 4.12)} \\
&= \left\| \overline{\mathbf{B}}[\mathrm{GEN}_0] - \overline{\mathbf{B}}[U_n] \right\|_{\max} && .
\end{aligned}$$

□

4.2 Shorter seed length via derandomized PRG products

In this section, we apply standard derandomization results (namely the INW generator) to reduce the seed length of the WPRG in Lemma 4.2. Specifically, we derandomize the products of PRGs $P_{i,1} \cdots P_{i,k}$ as follows.

Lemma 4.14. *Given $w \in \mathbb{N}$ and $\delta \in (0, 1/2)$ and a tuple of PRGs $M_1, \ldots, M_k$ where $M_i : \{0,1\}^s \to \{0,1\}^{l_i}$ and given i, M_i is computable in space $O(s)$, there exists an explicit PRG $\widetilde{M} : \{0,1\}^{\tilde{s}} \to \{0,1\}^l$ for $l = \sum_{i=1}^k l_i$ such that for every length-l, width-w ordered branching program $\mathbf{B}$, we have*

$$\|\overline{\mathbf{B}}[\widetilde{M}] - \overline{\mathbf{B}}[M_1 \cdots M_k]\|_{\max} \leq \delta$$

and $\widetilde{M}$ has seed length

$$\tilde{s} = s + O(\log k \cdot \log(kw/\delta)).$$

This result is obtained from recursive application of the derandomization lemma below. We view the Nisan PRG with seed length s composed with a branching program of width w as a branching program of width w, degree 2^s, and length 1. Thus, a product of these compositions can be derandomized using a PRG for branching programs of high degree. We use the formulation as stated in Lemma 11 of [13], which is an application of the INW generator.

Lemma 4.15 ([27])**.** *Let $G_1 : \{0,1\}^s \to \{0,1\}^{l_1}$ and $G_2 : \{0,1\}^s \to \{0,1\}^{l_2}$ be explicit PRGs. Then for every $\delta \in (0, 1/2)$ there is an explicit PRG $G : \{0,1\}^{s'} \to \{0,1\}^{l_1+l_2}$ where $s' = s + O(\log(w/\delta))$ such that for every pair of ordered branching programs $\mathbf{B}$, $\mathbf{B}'$ of width w and lengths l_1, l_2, respectively, we have*

$$\left\| (\overline{\mathbf{B}\mathbf{B}'})[G] - \overline{\mathbf{B}}[G_1]\overline{\mathbf{B}}'[G_2] \right\|_{\max} \leq \delta.$$

Given Lemma 4.14, we can reduce the seed length of all products of PRGs appearing in the WPRG of Lemma 4.2.

Corollary 4.16. *Fix $w \in \mathbb{N}$ and $\gamma \in (0, 1/2)$ and a family of length n WPRGs $\{\tau_i \cdot P_{i,1} \cdots P_{i,k} : i \in [V]\}$ where for all i, j, $\tau_i \in \{-1, 1\}$ and $P_{i,j}$ is a PRG with seed length s, and given i and j, the coefficient τ_i can be computed and the generator $P_{i,j}$ can be evaluated in space $O(s + \log V)$. Then there is an explicit $2V$-bounded WPRG GEN with seed length $s + O(\log k \cdot \log(wkV/\gamma))$ such that for every length-n, width-w branching program $\mathbf{B}$,*

$$\left\| \overline{\mathbf{B}}[\mathrm{GEN}] - \overline{\mathbf{B}}\left[\sum_{i\in[V]} \tau_i \cdot P_{i,1} \cdots P_{i,k}\right] \right\|_{\max} \leq \gamma.$$

Proof. For all $i \in [V]$, let GEN_i be the PRG obtained from applying Lemma 4.14 to $P_{i,1} \cdots P_{i,k}$ with $\delta = \gamma/V$. Then GEN_i is explicit and has seed length $s + O(\log k \cdot \log(wkV/\gamma))$. Finally, we apply Proposition 3.8 and define

$$\mathrm{GEN} = \sum_{i\in[V]} \tau_i \cdot \mathrm{GEN}_i \,.$$

Then for every length-n, width-w ordered branching program $\mathbf{B}$,

$$\begin{aligned}\left\| \overline{\mathbf{B}}[\mathrm{GEN}] - \overline{\mathbf{B}}\left[\sum_{i\in[V]} \tau_i \cdot P_{i_1} \cdots P_{i_k}\right] \right\|_{\max} &\leq \sum_{i\in[V]} \left\| \overline{\mathbf{B}}[\mathrm{GEN}_i] - \overline{\mathbf{B}}\left[P_{i,1} \cdots P_{i,k}\right] \right\|_{\max} \\ &\leq \frac{\gamma}{V} \cdot V\end{aligned}$$

and by Proposition 3.8 GEN is explicit and $2V$-bounded and has seed length

$$s + O(\log(V) + \log(k)\log(wkV/\gamma)) \,. \qquad \square$$

4.3 Putting it all together

We are now prepared to prove Theorem 4.1.

Theorem 4.1. *For all $n, w \in \mathbb{N}$ and $\varepsilon \in (0, 1/2)$, there exists an explicit ε-WPRG for the class of ordered branching programs of length n and width w with respect to $\|\cdot\|_{\max}$ with seed length*

$$s = O(\log(n)\log(nw) + \log(1/\varepsilon)\log\log_n(1/\varepsilon)).$$

Moreover, the generator is $\mathrm{poly}(1/\varepsilon)$ *bounded.*

Proof. We assume $\varepsilon = 1/n^{\Omega(1)}$ since otherwise the statement is satisfied by the Nisan PRG. Applying Lemma 4.2 with the same n, w and ε, we obtain a generator

$$\mathrm{GEN}_0 = \sum_{i\in[V]} \tau_i \cdot P_{i,1} P_{i,2} \cdots P_{i,k}$$

satisfying for every branching program $\mathbf{B}$ of length n and width w,

$$\left\|\overline{\mathbf{B}}[\mathrm{GEN}_0] - \overline{\mathbf{B}}[U_n]\right\|_{\max} \leq \varepsilon/2.$$

Furthermore, the family $\{\tau_i \cdot P_{i,1}P_{i,2}\cdots P_{i,k} : i \in [V]\}$ satisfies the requirements of Corollary 4.16 with $V = \mathrm{poly}(1/\varepsilon)$ and $k = O(\log_n(1/\varepsilon))$ and $s = O(\log n \cdot \log nw)$. Therefore, let GEN be the explicit WPRG obtained from applying Corollary 4.16 to this family with error $\gamma = \varepsilon/2$. Thus GEN is explicit, $2V = \mathrm{poly}(1/\varepsilon)$ bounded, and has seed length

$$s = O(\log(nw)\log(n) + \log(w/\varepsilon)\log\log_n(1/\varepsilon)).$$

As the seed length is greater than n otherwise, we can assume $\log(n) = \Omega(\log\log_n(1/\varepsilon))$ and ignore the $\log(w)\log\log_n(1/\varepsilon)$ term.

Finally, for every branching program $\mathbf{B}$ of length n and width w, we have

$$\begin{aligned}\left\|\overline{\mathbf{B}}[\mathrm{GEN}] - \overline{\mathbf{B}}[U_n]\right\|_{\max} &\leq \left\|\overline{\mathbf{B}}[\mathrm{GEN}] - \overline{\mathbf{B}}[\mathrm{GEN}_0]\right\|_{\max} + \left\|\overline{\mathbf{B}}[\mathrm{GEN}_0] - \overline{\mathbf{B}}[U_n]\right\|_{\max}\\ &\leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2}\end{aligned}$$

where the final line comes from our choice of error in Corollary 4.16 and Lemma 4.2. □

5 Pseudodistributions for permutation branching programs

In this section we prove Theorem 1.4. To do so, we restate it in the language of matrix-valued functions.

Theorem 5.1. *For all $n \in \mathbb{N}$ and $\varepsilon \in (0, 1/2)$, there exists an explicit ε-WPRG for the class of permutation branching programs of length n with respect to $\|\cdot\|_{\max}$ with seed length*

$$O(\log(n)\sqrt{\log(n/\varepsilon)}\sqrt{\log\log(n/\varepsilon)} + \log(1/\varepsilon)\log\log(n/\varepsilon)).$$

We prove Theorem 5.1 in a similar way to Theorem 4.1, with two major modifications. First, we use machinery from Ahmadinejad et al. [2] for a more sophisticated estimate of the norm of the error matrix $\mathbf{Err}$. Second, we use tools from Hoza, Pyne and Vadhan [26] to derandomize concatenations of WPRGs in a way that avoids dependence on the width of the branching programs being fooled.

5.1 A WPRG with large seed length

For the duration of the section we will assume that $n + 1$ is a power of two. This is without loss of generality, as any prefix of an ε-WPRG for permutation branching programs must also ε-fool permutation branching programs, as the final layers could be the identity.

In the next few subsections we prove the following analogue of Lemma 4.2.

Theorem 5.2. *Given $n \in \mathbb{N}$ and $\varepsilon, \delta \in (0, 1/2)$, let $\ell = O(\log_{1/\delta}(n/\varepsilon))$. Then there exists an explicit weighted generator GEN_0 such that GEN_0 is ε-pseudorandom for the class of permutation branching programs of length n and arbitrary width with respect to $\|\cdot\|_{\max}$ and*

$$\mathrm{GEN}_0 = \sum_{i \in [V]} \tau_i \cdot P_{i,1} P_{i,2} \cdots P_{i,k}$$

such that

1. $V = n^{O(\ell)}$
2. $k = O(\ell \cdot \log n)$
3. *For all i, $\tau_i \in \{-1, 1\}$.*
4. *For all i, j, $P_{i,j}$ is an (unweighted) PRG with seed length $s = O(\log n \cdot \log(\log(n)/\delta))$.*
5. *Given $i \in [V]$ and $j \in [k]$, τ_i and $P_{i,j}$ are evaluatable in space $O(s + \log V)$.*

Applying the sum and product rules to the output of the lemma, we obtain an ε-WPRG for permutation branching programs with one accept vertex with seed length $\omega(\ell \cdot \log^2 n)$, worse than the PRG of [26]. However, since $\ell \cdot \log n$ is only polylogarithmic in n for our choice of δ, we can apply derandomization results of [26] to shrink the seed length of each summand.

5.2 The lift transition matrix

Given a branching program $\mathbf{B}$, we define a matrix-valued function that holds information about transitions between all pairs of layers.

Definition 5.3. Given $n, w \in \mathbb{N}$ and a length-n, width-w permutation branching program $\mathbf{B}$, define the **lift transition matrix** $\mathbf{B} : \{0,1\} \to \mathbb{R}^{(n+1)w \times (n+1)w}$ by

$$\mathbf{B}[s] = \begin{bmatrix} 0 & \mathbf{B}_{0..1}[s] & 0 & \dots & 0 \\ 0 & 0 & \mathbf{B}_{1..2}[s] & & \vdots \\ \vdots & & & \ddots & 0 \\ 0 & & & 0 & \mathbf{B}_{n-1..n}[s] \\ \mathbf{J}_w & 0 & \dots & & 0 \end{bmatrix}$$

where we view the output as an $(n+1) \times (n+1)$ block matrix. For all $i \in \mathbb{N}$, define $\mathbf{B}^{(i)} : \{0,1\}^i \to \mathbb{R}^{(n+1)w \times (n+1)w}$ as $\mathbf{B}^{(i)}[s_1 \dots s_i] = \mathbf{B}[s_1]\mathbf{B}[s_2] \cdots \mathbf{B}[s_i]$.

We start by analyzing the structure of these matrices.

Proposition 5.4. *Let $\mathbf{B}$ be the lift transition matrix for a length-n, width-w permutation branching program. For all $x \in \mathbb{N}$, $s \in \{0,1\}^x$ and $j, k \in \{0, \dots, n\}$, let $m = j + x \mod n+1$. Then we have*

$$(\mathbf{B}^{(x)}[s])_{j,k} = \begin{cases} 0 & k \neq m \\ \mathbf{J}_w & m \leq j \text{ or } x > n \\ \mathbf{B}_{j..k}[s] & j < m \text{ and } x \leq n \end{cases}$$

Proof. The only nonzero blocks of $\mathbf{B}$ have index $(i, i+1 \mod n+1)$. Thus,

$$\begin{aligned}\left(\mathbf{B}^{(x)}[s]\right)_{j,k} &= \sum_{j_1..j_{x-1}} \mathbf{B}[s_1]_{j,j_1}\mathbf{B}[s_2]_{j_1,j_2}\cdots\mathbf{B}[s_x]_{j_{x-1},k} \\ &= \mathbf{B}[s_1]_{j,j+1}\mathbf{B}[s_2]_{j+1,j+2}\cdots\mathbf{B}[s_x]_{j+x-1,k}\end{aligned}$$

where all block indices are written mod $n+1$.

Clearly if $k \neq j + x \mod n+1$ this is zero. Then if $\mathbf{B}[s_i]_{n,0} = \mathbf{J}_w$ appears in this product we have $(\mathbf{B}^{(x)}[s])_{j,k} = \mathbf{J}_w$, as $\mathbf{A} \cdot \mathbf{J}_w = \mathbf{J}_w \cdot \mathbf{A} = \mathbf{J}_w$ for every doubly stochastic matrix $\mathbf{A}$, and $\mathbf{B}[b]_{j,j+1}$ is doubly stochastic for all j, b. Otherwise the product is of the form $\mathbf{B}_{j..j+1}[s_1]\mathbf{B}_{j+1..j+2}[s_2]\cdots\mathbf{B}_{j+x-1,j+x}[s_x] = \mathbf{B}_{j..j+x}[s]$ as desired. ◻

Note that when $x > n$, $\mathbf{B}^{(x)}[s]$ has no dependence on the branching program $\mathbf{B}$ or the input s; all nonzero blocks equal $\mathbf{J}_w$ and the location of those blocks depends only on x and n. In particular, we have the following corollary.

Corollary 5.5. *Let* $\mathbf{B}$ *be the lift transition matrix for a length-n, width-w permutation branching program and let* $F : \{0,1\}^s \to \{0,1\}^x, G : \{0,1\}^{s'} \to \{0,1\}^x$ *be arbitrary PRGs. If* $x > n$, *then* $\overline{\mathbf{B}^{(x)}}[F] = \overline{\mathbf{B}^{(x)}}[G]$.

Proof. By Proposition 5.4 the nonzero blocks of $\overline{\mathbf{B}^{(x)}}[F]$ and $\overline{\mathbf{B}^{(x)}}[G]$ are located only at indices $i, i+x \mod n+1$ and are all equal to $\mathbf{J}_w$. ◻

If $x < n$, we can eliminate the dependence on the generator by multiplying by $(\mathbf{I}_{n+1} \otimes \mathbf{J}_w)$.

Corollary 5.6. *Let* $\mathbf{B}$ *be the lift transition matrix for a length-n, width-w permutation branching program and let* $F : \{0,1\}^s \to \{0,1\}^x, G : \{0,1\}^{s'} \to \{0,1\}^x$ *be arbitrary PRGs. Then*

$$(\mathbf{I}_{n+1} \otimes \mathbf{J}_w)\overline{\mathbf{B}^{(x)}}[F] = (\mathbf{I}_{n+1} \otimes \mathbf{J}_w)\overline{\mathbf{B}^{(x)}}[G]\,.$$

Proof. By Proposition 5.4 the nonzero blocks of $\overline{\mathbf{B}^{(x)}}[F]$ and $\overline{\mathbf{B}^{(x)}}[G]$ are located only at indices $i, i+x \mod n+1$ and these blocks are convex combinations of doubly stochastic matrices and are thus doubly stochastic, so the result follows from the fact that $\mathbf{J}_w \cdot \mathbf{A} = \mathbf{J}_w$ for every doubly stochastic matrix $\mathbf{A}$. ◻

These corollaries will enable long outputs to exactly cancel in the error-reduction procedure we give later.

5.3 Approximating powers

To analyze the distribution of PRGs over these transition matrices, we introduce the idea of a cyclic branching program, and recall a consequence of [26].

Definition 5.7. A length-n, width-w permutation branching program B is **cyclic** if it has transition functions $B_1, \ldots, B_n, B_0$, where B_0 is a transition function from layer n to layer 0. Given a cyclic branching program B, define the **cyclic transition matrix** as the function $\mathbf{B} : \{0,1\} \to \mathbb{R}^{(n+1)w \times (n+1)w}$ where

$$\mathbf{B}[s] = \begin{bmatrix} 0 & \mathbf{B}_{0..1}[s] & 0 & \ldots & 0 \\ 0 & 0 & \mathbf{B}_{1..2}[s] & & \vdots \\ \vdots & & & \ddots & 0 \\ 0 & & & 0 & \mathbf{B}_{n-1..n}[s] \\ \mathbf{B}_{n..0}[s] & 0 & \ldots & & 0 \end{bmatrix}.$$

Furthermore, for all i define $\mathbf{B}^{(i)}[s_1 \ldots s_i] = \mathbf{B}[s_1] \ldots \mathbf{B}[s_i]$.

We then state a convenient form of the main theorem of [26]. To do so, we review the notion of approximation introduced by Ahmadinejad et al. [2], which plays a central role in their analysis. 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. Note that our use of row vectors means the right vector has the conjugate transpose applied, where in [2] this is reversed.

Definition 5.8 (Unit-Circle Approximation [2])**.** For $\mathbf{A}, \widetilde{\mathbf{A}} \in \mathbb{C}^{N \times N}$ and $\varepsilon \geq 0$, we say $\mathbf{A}$ is an ε-**unit-circle approximation** of $\widetilde{\mathbf{A}}$, denoted $\mathbf{A} \overset{\circ}{\approx}_\varepsilon \widetilde{\mathbf{A}}$, if

$$\forall x, y \in \mathbb{C}^N, \quad \left| x(\mathbf{A} - \widetilde{\mathbf{A}})y^* \right| \leq \frac{\varepsilon}{2} \left(\|x\|^2 + \|y\|^2 - \left| x\widetilde{\mathbf{A}}x^* + y\widetilde{\mathbf{A}}y^* \right| \right).$$

One nice feature of unit circle approximation is that it is preserved under convex combinations.

Proposition 5.9. *Let $\mathbf{A}, \widetilde{\mathbf{A}}, \mathbf{X}, \widetilde{\mathbf{X}} \in \mathbb{C}^{N \times N}$ where $\mathbf{A} \overset{\circ}{\approx}_\varepsilon \widetilde{\mathbf{A}}$ and $\mathbf{X} \overset{\circ}{\approx}_\varepsilon \widetilde{\mathbf{X}}$. Then for every $\lambda \in [0,1]$, $\lambda\mathbf{A} + (1-\lambda)\mathbf{X} \overset{\circ}{\approx}_\varepsilon \lambda\widetilde{\mathbf{A}} + (1-\lambda)\widetilde{\mathbf{X}}$.*

Proof. Fix arbitrary $\forall x, y \in \mathbb{C}^N$, then

$$\begin{aligned} &\left| x(\lambda\mathbf{A} + (1-\lambda)\mathbf{X} - (\lambda\widetilde{\mathbf{A}} + (1-\lambda)\widetilde{\mathbf{X}}))y^* \right| \\ &\quad \leq \lambda \left| x(\mathbf{A} - \widetilde{\mathbf{A}})y^* \right| + (1-\lambda) \left| x(\mathbf{X} - \widetilde{\mathbf{X}})y^* \right| \\ &\quad \leq \frac{\varepsilon}{2} \left(\|x\|_2^2 + \|y\|_2^2 - \lambda \left| x\widetilde{\mathbf{A}}x^* + y\widetilde{\mathbf{A}}y^* \right| - (1-\lambda) \left| x\widetilde{\mathbf{X}}x^* + y\widetilde{\mathbf{X}}y^* \right| \right) \\ &\quad \leq \frac{\varepsilon}{2} \left(\|x\|_2^2 + \|y\|_2^2 - \left| x(\lambda\widetilde{\mathbf{A}} + (1-\lambda)\widetilde{\mathbf{X}})x^* + y(\lambda\widetilde{\mathbf{A}} + (1-\lambda)\widetilde{\mathbf{X}})y^* \right| \right). \end{aligned}$$ □

As a consequence, multiplying by subunit scalars preserves unit circle approximation.

Corollary 5.10. *For $\mathbf{A}, \widetilde{\mathbf{A}} \in \mathbb{C}^{N \times N}$, if $\mathbf{A} \overset{\circ}{\approx}_\varepsilon \widetilde{\mathbf{A}}$, for all $c \in \mathbb{R}$ with $c \in [0,1]$ we have $c\mathbf{A} \overset{\circ}{\approx}_\varepsilon c\widetilde{\mathbf{A}}$.*

Proof. This follows from Proposition 5.9, taking $\lambda = c$ and $\mathbf{X} = \widetilde{\mathbf{X}} = 0$. □

We can then recall the consequence of [26]. The result uses that cyclic transition matrices correspond to transition matrices of consistently-labeled graphs, and the correspondence between the INW generator and the derandomized square of Rozenman and Vadhan [41].

Theorem 5.11 (Consequence of [26] Theorem 1.4)**.** *For all $q \in \mathbb{N}$ and $\varepsilon \in (0, 1/2)$, there is a family of explicit PRGs $\mathrm{INW}_0, \ldots, \mathrm{INW}_q$ such that INW_0 is the trivial PRG on one bit, and for all $i \in [q]$, INW_i has seed length $s = O(q \cdot \log(1/\varepsilon))$ and produces 2^i bits of output. Furthermore, for all $i \in \{0, \ldots, q-1\}$, for every cyclic permutation branching program B with cyclic transition matrix $\mathbf{B}$,*

$$\overline{\mathbf{B}^{(2^{i+1})}}\,[\mathrm{INW}_{i+1}] \overset{\circ}{\approx}_{\varepsilon} \overline{\mathbf{B}^{(2^{i+1})}}\,[\mathrm{INW}_i \cdot \mathrm{INW}_i]\,.$$

We can then write the lift transition matrix of a permutation branching program as the convex combination of cyclic transition matrices, and by doing so obtain a bound of the same form as Theorem 5.11.

Definition 5.12. For all $n \in \mathbb{N}$ and $\delta \in (0, 1/2)$, let $\mathrm{INW}_0, \ldots, \mathrm{INW}_q$ be the family of PRGs obtained from applying Theorem 5.11 with $q = \log(n+1)$ and error $\delta = \delta/40q^2$. These generators have seed length $s = O(q \log(q/\delta))$. Then for every length-n permutation branching program $\mathbf{B}$ with lift transition matrix $\mathbf{B}$, for all $i \in \{0, \ldots, q\}$ define

$$\mathbf{W}_i = \overline{\mathbf{B}^{(2^i)}}\,[\mathrm{INW}_i]\,.$$

We remark that since INW_0 is the trivial PRG on one bit, $\mathbf{W}_0$ is defined identically to what is stated in Subsection 2.2. We now show that these matrices successively approximate each other with respect to unit circle approximation.

Lemma 5.13. *Given $n \in \mathbb{N}$ and $\delta \in (0, 1/2)$ and a length-n permutation branching program $\mathbf{B}$, for all $i \in \{0, \ldots, q\}$ let $\mathbf{W}_i$ be as defined as in Definition 5.12 with the same n, δ. Then for all $i \in [q]$,*

$$\mathbf{W}_i \overset{\circ}{\approx}_{\frac{\delta}{40q^2}} \mathbf{W}_{i-2}^2.$$

Proof. For all $y \in [w]$ let $\mathbf{A}_y$ be the cyclic transition matrix of the length-n cyclic permutation branching program A_y, which has transition functions $B_1, \ldots, B_n, A_0^y$ where $A_0^y(v, b) = (v - 1 + y \mod w) + 1$. Observe that with our choice of A_0^y, all walks passing from layer n to layer 0 are distributed uniformly over the relevant set of end states, where the distribution is over y. We first claim $\frac{1}{w}\sum_{y=1}^{w} \mathbf{A}_y[s] = \mathbf{B}[s]$. Fixing arbitrary s, $(\mathbf{A}_y[s])_{i,j} = \mathbf{B}[s]_{i,j}$ for all $y \in [w]$ and all blocks $(i, j) \neq (n, 0)$. For the $(n, 0)$ block, for all $u, v \in \{0, \ldots, w-1\}$,

$$\left(\left(\frac{1}{w}\sum_{y=1}^{w} \mathbf{A}_y[s]\right)_{n,0}\right)_{u,v} = \Pr_{y \in [w]}[u + y \mod w = v] = \frac{1}{w} = (\mathbf{J}_w)_{u,v}.$$

Furthermore by Theorem 5.11, for all $y \in [w]$ and $i \in [q]$ we have

$$\overline{\mathbf{A}_y^{(2^i)}}[\mathrm{INW}_i] \overset{\circ}{\approx}_{\delta/40q^2} \overline{\mathbf{A}_y^{(2^i)}}[\mathrm{INW}_{i-1} \cdot \mathrm{INW}_{i-1}]$$

so we obtain

$$\begin{aligned}
\mathbf{W}_i &= \overline{\mathbf{B}^{(2^i)}}[\mathrm{INW}_i] \\
&= \frac{1}{w}\sum_{y=1}^{w} \overline{\mathbf{A}_y^{(2^i)}}[\mathrm{INW}_i] \\
&\overset{\circ}{\approx}_{\delta/40q^2} \frac{1}{w}\sum_{y=1}^{w} \overline{\mathbf{A}_y^{(2^i)}}[\mathrm{INW}_{i-1}\cdot \mathrm{INW}_{i-1}] && \text{(Proposition 5.9)} \\
&= \overline{\mathbf{B}^{(2^i)}}[\mathrm{INW}_{i-1}\cdot \mathrm{INW}_{i-1}] \\
&= \mathbf{W}_{i-1}^2
\end{aligned}$$

□

5.4 The cycle-lift Laplacian

We wish to use this sequence of approximations $\mathbf{W}_0,\dots,\mathbf{W}_q$ to construct a good approximation to $\mathbf{W}_0^n$. To do so, we slightly modify the construction described in Section 6 of [2]. They tensor the matrix to be approximated with a cycle of sufficient length (in our case $n+1$). To do so, they first define a series of cycles. Let $\mathbf{C}_i$ be the directed cycle on 2^i vertices. Without loss of generality, we use the following ordering of rows and columns of $\mathbf{C}$.

Definition 5.14. Let $\mathbf{C}_0 = [1]$ and given $\mathbf{C}_i$ let $\mathbf{C}_{i+1} = \begin{bmatrix} 0 & \mathbf{I}_{2^i} \\ \mathbf{C}_i & 0 \end{bmatrix}$.

Now let $\pi_r : \{0,\dots,2^r-1\} \to \{0,\dots,2^r-1\}$ be the bijection from the usual ordering of the 2^r-cycle to the indexing in $\mathbf{C}_r$. Specifically, writing $u \in \{0,\dots,2^r-1\}$ in binary as $u = u_{r-1}\dots u_0$, we have $\pi_r(u) = u_0\dots u_{r-1} = u_0\cdot 2^{r-1} + \pi_{r-1}(u_{r-1}\dots u_1)$. From this, we can relate the indexing of block submatrices to that of a larger matrix.

Claim 5.15. *Let* $\mathbf{M} = \begin{bmatrix} \mathbf{A}_{0,0} & \mathbf{A}_{0,1} \\ \mathbf{A}_{1,0} & \mathbf{A}_{1,1} \end{bmatrix}$ *be a* $2^r \times 2^r$ *matrix. Then for every* $u, v \in \{0,\dots,2^{r-1}-1\}$ *and* $b, c \in \{0,1\}$, *we have* $\mathbf{M}_{\pi_r(2u+b),\pi_r(2v+c)} = (\mathbf{A}_{bc})_{\pi_{r-1}(u),\pi_{r-1}(v)}$.

Proof. Using the definition of the bijection π_r, we have that $\pi_r(2u+b) = b\cdot 2^{r-1} + \pi_{r-1}(u)$, which is precisely equivalent to selecting a block via b and the row index of the $2^{r-1}\times 2^{r-1}$ submatrix via $\pi_{r-1}(u)$, and the same holds for the column $\pi_r(2v+c)$, so the claim follows. □

We now take a Laplacian of $\mathbf{C}_q \otimes \mathbf{W}_0$ where $q = \log(n+1)$.

Definition 5.16. Given $n \in \mathbb{N}$, $\delta \in (0,1/2)$ and a length-n permutation branching program $\mathbf{B}$ with lift transition matrix $\mathbf{B}$, for all $i \in \{0,\dots,q\}$ let $\mathbf{W}_i$ be defined as in Definition 5.12 with the same n, δ. For convenience we define $N = (n+1)w$ for the remainder of the section. Then fix $c = 1 - 1/(n+1)$ and define the **cycle-lift Laplacian** of $\mathbf{B}$ as

$$\mathbf{L} = \mathbf{L}^{(0)} = \mathbf{I}_{2^q N} - \mathbf{C}_q \otimes c\mathbf{W}_0.$$

For convenience, we write $c_i = c^{2^i}$. This definition is identical to that in [2] except we multiply $\mathbf{W}_0$ by a factor strictly less than 1. This makes $\mathbf{L}$ invertible, making the analysis cleaner and providing a bound on the singular values of $\mathbf{L}$ that is independent of the width of the branching program, which enables us to obtain a seed length independent of width. We now detail the decomposition of this Laplacian using repeated Schur complements, identical to that of Section 6 of [2].

Let H_q be the set of indices $\{2^{q-1}N, 2^{q-1}N+1, \ldots, 2^qN-1\}$, so that the cycle $\mathbf{C}_q$ alternates between the indices in H_q and those in H_q^c. The Schur complement of $\mathbf{L}$ onto the indices H_q is an $|H_q| \times |H_q|$ matrix which is shown in [2] to have the following nice form.

$$\mathrm{Sc}(\mathbf{L}, H_q) = \mathbf{I}_{2^{q-1}N} - \mathbf{C}_{q-1} \otimes c^2\mathbf{W}_0^2$$

This is exactly the Laplacian of the cycle lift of $c^2\mathbf{W}_0^2$ with $\mathbf{C}_{q-1}$. We then replace $\mathbf{W}_0^2$ with $\mathbf{W}_1$ and again take the Schur complement with respect to H_{q-1}. We repeat this procedure q times to obtain a decomposition of the original Laplacian $\mathbf{L}$ in terms of repeated Schur complements. Formally, define $\mathbf{L}^{(0)} = \mathbf{L}$ and for all $i \in [q]$ define

$$\mathbf{L}^{(i)} = \mathbf{X}_1 \ldots \mathbf{X}_i \begin{bmatrix} \mathbf{I}_{(2^q-2^{q-i})N} & 0 \\ 0 & \mathbf{I}_{2^{q-i}N} - \mathbf{C}_{q-i} \otimes c_i\mathbf{W}_i \end{bmatrix} \mathbf{Y}_i \ldots \mathbf{Y}_1 \tag{5.1}$$

where

$$\mathbf{X}_j = \begin{bmatrix} \mathbf{I}_{(2^q-2^{q-j+1})N} & 0 & 0 \\ 0 & \mathbf{I}_{2^{q-j}N} & 0 \\ 0 & -\mathbf{C}_{q-j} \otimes c_{j-1}\mathbf{W}_{j-1} & \mathbf{I}_{2^{q-j}N} \end{bmatrix}, \mathbf{Y}_j = \begin{bmatrix} \mathbf{I}_{(2^q-2^{q-j+1})N} & 0 & 0 \\ 0 & \mathbf{I}_{2^{q-j}N} & -\mathbf{I}_{2^{q-j}} \otimes c_{j-1}\mathbf{W}_{j-1} \\ 0 & 0 & \mathbf{I}_{2^{q-j}N}\,. \end{bmatrix}$$

Later we will argue that all these matrices $\mathbf{L}^{(i)}$are good approximations of $\mathbf{L}$ (in an appropriate sense). Moreover, a computation shows that the inverse of $\mathbf{L}^{(q)}$ is

$$(\mathbf{L}^{(q)})^{-1} = \mathbf{Y}_1^{-1} \ldots \mathbf{Y}_q^{-1} \begin{bmatrix} \mathbf{I}_{(2^q-1)N} & 0 \\ 0 & (\mathbf{I}_N - \mathbf{C}_0 \otimes c_q\mathbf{W}_q)^{-1} \end{bmatrix} \mathbf{X}_q^{-1} \ldots \mathbf{X}_1^{-1} \tag{5.2}$$

where

$$\mathbf{X}_j^{-1} = \begin{bmatrix} \mathbf{I}_{(2^q-2^{q-j+1})N} & 0 & 0 \\ 0 & \mathbf{I}_{2^{q-j}N} & 0 \\ 0 & \mathbf{C}_{q-j} \otimes c_{j-1}\mathbf{W}_{j-1} & \mathbf{I}_{2^{q-j}N} \end{bmatrix}, \mathbf{Y}_j^{-1} = \begin{bmatrix} \mathbf{I}_{(2^q-2^{q-j+1})N} & 0 & 0 \\ 0 & \mathbf{I}_{2^{q-j}N} & \mathbf{I}_{2^{q-j}} \otimes c_{j-1}\mathbf{W}_{j-1} \\ 0 & 0 & \mathbf{I}_{2^{q-j}N} \end{bmatrix}.$$

Note that it is easy to invert the final block diagonal matrix at the bottom of the recursion, as

$$(\mathbf{I}_N - \mathbf{C}_0 \otimes c_q\mathbf{W}_q)^{-1} = (\mathbf{I}_N - c_q\mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} = \mathbf{I}_N + \left(\sum_{i=1}^{\infty} c_q^i\right)(\mathbf{I}_{n+1} \otimes \mathbf{J}_w)$$

where the first equality holds because $\mathbf{W}_q = \overline{\mathbf{B}^{(2^q)}}\left[\mathrm{INW}_q\right] = \mathbf{I}_{n+1} \otimes \mathbf{J}_w$ (by Corollary 5.6) and the second because $c_q < 1$. Note that this matrix has no dependence on the branching program B.

We now describe the form of the blocks of $(\mathbf{L}^{(q)})^{-1}$ in terms of the base matrices $\mathbf{W}_i$. To do so, we define $\mathbf{D}_r$ for $r \in \{0, \ldots, q\}$. The rows and columns of $\mathbf{D}_r$ correspond to vertices of the original cycle $\mathbf{C}_q$.

Definition 5.17. Let $\mathbf{D}_0 = (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} \in \mathbb{R}^{N \times N}$. Given $\mathbf{D}_r \in \mathbb{R}^{2^r N \times 2^r N}$, define $\mathbf{D}_{r+1} \in \mathbb{R}^{2^{r+1} N \times 2^{r+1} N}$ as

$$\mathbf{D}_{r+1} = \begin{bmatrix} \mathbf{I}_{2^r N} + (\mathbf{I}_{2^r} \otimes c_{q-r-1}\mathbf{W}_{q-r+1})\mathbf{D}_r(\mathbf{C}_r \otimes c_{q-r-1}\mathbf{W}_{q-r-1}) & (\mathbf{I}_{2^r} \otimes c_{q-r-1}\mathbf{W}_{q-r+1})\mathbf{D}_r \\ \mathbf{D}_r(\mathbf{C}_r \otimes c_{q-r-1}\mathbf{W}_{q-r-1}) & \mathbf{D}_r \end{bmatrix}.$$

Lemma 5.18. *We have* $\mathbf{D}_q = (\mathbf{L}^{(q)})^{-1}$.

Proof. We show by induction that from $r = 0$ up to q that

$$(\mathbf{L}^{(q)})^{-1} = \mathbf{Y}_1^{-1} \ldots \mathbf{Y}_{q-r}^{-1} \begin{bmatrix} \mathbf{I}_{(2^q - 2^r)N} & 0 \\ 0 & \mathbf{D}_r \end{bmatrix} \mathbf{X}_{q-r}^{-1} \ldots \mathbf{X}_1^{-1}.$$

This is immediate for $\mathbf{D}_0$. Then assuming it holds for $\mathbf{D}_r$, we have

$$\begin{aligned}
(\mathbf{L}^{(q)})^{-1} &= \mathbf{Y}_1^{-1} \ldots \mathbf{Y}_{q-r}^{-1} \begin{bmatrix} \mathbf{I}_{(2^q - 2^r)N} & 0 \\ 0 & \mathbf{D}_r \end{bmatrix} \mathbf{X}_{q-r}^{-1} \ldots \mathbf{X}_1^{-1} \\
&= \mathbf{Y}_1^{-1} \ldots \mathbf{Y}_{q-r}^{-1} \begin{bmatrix} \mathbf{I}_{(2^q - 2^{r+1})N} & 0 & 0 \\ 0 & \mathbf{I}_{2^r N} & 0 \\ 0 & 0 & \mathbf{D}_r \end{bmatrix} \mathbf{X}_{q-r}^{-1} \ldots \mathbf{X}_1^{-1} \\
&= \mathbf{Y}_1^{-1} \ldots \mathbf{Y}_{q-r}^{-1} \begin{bmatrix} \mathbf{I}_{(2^q - 2^{r+1})N} & 0 & 0 \\ 0 & \mathbf{I}_{2^r N} & 0 \\ 0 & \mathbf{D}_r(\mathbf{C}_r \otimes c_{q-r-1}\mathbf{W}_{q-r-1}) & \mathbf{D}_r \end{bmatrix} \mathbf{X}_{q-r-1}^{-1} \ldots \mathbf{X}_1^{-1} \\
&= \mathbf{Y}_1^{-1} \ldots \mathbf{Y}_{q-r-1}^{-1} \begin{bmatrix} \mathbf{I}_{(2^q - 2^{r+1})N} & 0 \\ 0 & \mathbf{D}_{r+1} \end{bmatrix} \mathbf{X}_{q-r-1}^{-1} \ldots \mathbf{X}_1^{-1}
\end{aligned}$$ □

We now show that the blocks of $\mathbf{D}_q = (\mathbf{L}^{(q)})^{-1}$ consist essentially of products of some of the $\mathbf{W}_i$.

Lemma 5.19. *For all* $r \in \{0, \ldots, q\}$ *and* $u, v \in \{0, \ldots, 2^r - 1\}$*, let* $m = 2^{q-r}(v - u \mod 2^r)$*. Then*

$$(\mathbf{D}_r)_{\pi_r(u), \pi_r(v)} = (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^m \mathbf{W}_{i_1} \ldots \mathbf{W}_{i_k}$$

for some indices $i_1, \ldots, i_k$ *such that* $\sum_{j=1}^k 2^{i_j} = m$ *and* $k \le 2r$*. Furthermore, given* u, v *the indices* $i_1, \ldots, i_k$ *are computable in space* $O(\log n)$*.*

Proof. We prove this using induction from $r = 0$ up to q. For the base case, there is only one block $u = v = 0$ and $(\mathbf{D}_0)_{\pi_0(0), \pi_0(0)} = (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1}$ as claimed. Now assume that $\mathbf{D}_r$ has the claimed form.

Let $u', v' \in \{0, \ldots, 2^{r+1} - 1\}$ be arbitrary indices in the index set of $\mathbf{D}_{r+1}$. We verify $(\mathbf{D}_{r+1})_{\pi_{r+1}(u'), \pi_{r+1}(v')}$ has the claimed form via casework. In all cases we take $u, v \in \{0, \ldots, 2^r - 1\}$.

1. If $u' = 2u + 1$ and $v' \in 2v + 1$, we have

$$\begin{aligned}(\mathbf{D}_{r+1})_{\pi_{r+1}(u'),\pi_{r+1}(v')} &= (\mathbf{D}_r)_{\pi_r(u),\pi_r(v)} \\ &= (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^l \widetilde{\mathbf{W}}\end{aligned}$$

   where the first line follows from Definition 5.17 and Claim 5.15. From the inductive assumption, $\widetilde{\mathbf{W}}$ is a product of matrices $\mathbf{W}_{i_j}$ with total length

$$\begin{aligned}l &= 2^{q-r}(v - u \mod 2^r) \\ &= 2^{q-r-1}(2v - 1 - (2u - 1) \mod 2^{r+1}) \\ &= 2^{q-r-1}(v' - u' \mod 2^{r+1})\end{aligned}$$

   as desired.

2. If $u' = 2u + 1$ and $v' = 2v$, we have

$$\begin{aligned}(\mathbf{D}_{r+1})_{\pi_{r+1}(u'),\pi_{r+1}(v')} &= (\mathbf{D}_r(\mathbf{C}_r \otimes c_{q-r-1}\mathbf{W}_{q-r-1}))_{\pi_r(u),\pi_r(v)} \\ &= (\mathbf{D}_r)_{\pi_r(u),\pi_r(v-1)} c_{q-r-1}\mathbf{W}_{q-r-1} \\ &= (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^l \widetilde{\mathbf{W}} c_{q-r-1}\mathbf{W}_{q-r-1}\end{aligned}$$

   where the second line follows from the fact that $(\mathbf{D}_r(\mathbf{C}_r \otimes \mathbf{I}_N))_{\pi(a),\pi(b)} = (\mathbf{D}_r)_{\pi_r(a),\pi_r(b-1)}$ for all a, b. From the inductive assumption, we have that the product of matrices has length

$$\begin{aligned}l + 2^{q-r-1} &= 2^{q-r}(v - 1 - u \mod 2^r) + 2^{q-r-1} \\ &= 2^{q-r-1}(2v - (2u + 1) \mod 2^{r+1}) \\ &= 2^{q-r-1}(v' - u' \mod 2^{r+1})\end{aligned}$$

   as desired.

3. If $u' = 2u$ and $v' = 2v + 1$, we have

$$\begin{aligned}(\mathbf{D}_{r+1})_{\pi_{r+1}(u'),\pi_{r+1}(v')} &= ((\mathbf{I}_{2^r} \otimes c_{q-r-1}\mathbf{W}_{q-r+1})\mathbf{D}_r)_{\pi_r(u),\pi_r(v)} \\ &= c_{q-r-1}\mathbf{W}_{q-r-1}(\mathbf{D}_r)_{\pi_r(u),\pi_r(v)} \\ &= c_{q-r-1}\mathbf{W}_{q-r-1}(\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^l \widetilde{\mathbf{W}} \\ &= c_{q-r-1}(\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^l \mathbf{W}_{q-r-1}\widetilde{\mathbf{W}} \qquad \text{(Corollary 5.6)}\end{aligned}$$

   From the inductive assumption, we have that the product of matrices has length

$$\begin{aligned}l + 2^{q-r-1} &= 2^{q-r}(v - u \mod 2^r) + 2^{q-r-1} \\ &= 2^{q-r-1}(2v + 1 - 2u \mod 2^{r+1}) \\ &= 2^{q-r-1}(v' - u' \mod 2^{r+1})\end{aligned}$$

   as desired.

4. If $u' = 2u$ and $v' = 2v$, we do casework based on if $u' = v'$, due to the presence of $\mathbf{I}_{2^r N}$ in the relevant quadrant of $\mathbf{D}_{r+1}$. If $u' \neq v'$, we have

$$\begin{aligned}(\mathbf{D}_{r+1})_{\pi_{r+1}(u'),\pi_{r+1}(v')} &= ((\mathbf{I}_{2^r} \otimes c_{q-r-1}\mathbf{W}_{q-r+1})\mathbf{D}_r(\mathbf{C}_r \otimes c_{q-r-1}\mathbf{W}_{q-r-1}))_{\pi_r(u),\pi_r(v)} \\ &= c_{q-r-1}\mathbf{W}_{q-r-1}(\mathbf{D}_r)_{\pi_r(u),\pi_r(v-1)}c_{q-r-1}\mathbf{W}_{q-r-1} \\ &= (\mathbf{I}_N - c_q\mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1}c^l c_{q-r}\mathbf{W}_{q-r-1} \cdot \widetilde{\mathbf{W}} \cdot \mathbf{W}_{q-r-1}\end{aligned}$$

where the second line follows from identical reasoning to Case 2. From the inductive assumption, we have that the product of matrices has length

$$\begin{aligned}l + 2^{q-r} &= 2^{q-r}(v - 1 - u \mod 2^r) + 2^{q-r} \\ &= 2^{q-r-1}(2v - 2u \mod 2^{r+1}) \\ &= 2^{q-r-1}(v' - u' \mod 2^{r+1})\end{aligned}$$

as desired.

Finally, if $u' = v'$ we have

$$\begin{aligned}(\mathbf{D}_{r+1})_{\pi_{r+1}(u'),\pi_{r+1}(v')} &= (\mathbf{I}_{2^r N} + (\mathbf{I}_{2^r} \otimes c_{q-r-1}\mathbf{W}_{q-r+1})\mathbf{D}_r(\mathbf{C}_r \otimes c_{q-r-1}\mathbf{W}_{q-r-1}))_{\pi_r(u),\pi_r(v)} \\ &= \mathbf{I}_N + c_{q-r-1}\mathbf{W}_{q-r-1}(\mathbf{D}_r)_{\pi_r(v),\pi_r(v-1)}c_{q-r-1}\mathbf{W}_{q-r-1} \\ &= \mathbf{I}_N + (\mathbf{I}_N - c_q\mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1}c^l c_{q-r}\mathbf{W}_{q-r-1} \cdot \widetilde{\mathbf{W}} \cdot \mathbf{W}_{q-r-1}\end{aligned}$$

Here we have that the product of matrices has length

$$\begin{aligned}l + 2 \cdot 2^{q-r-1} &= 2^{q-r}(v - 1 - v \mod 2^r) + 2^{q-r} \\ &= 2^{q-r}(2^r - 1) + 2^{q-r} = 2^q\end{aligned}$$

so we obtain

$$\begin{aligned}(\mathbf{D}_{r+1})_{\pi_{r+1}(u'),\pi_{r+1}(v')} &= \mathbf{I}_N + (\mathbf{I}_N - c_q\mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1}c^l c_{q-r}\mathbf{W}_{q-r-1} \cdot \widetilde{\mathbf{W}} \cdot \mathbf{W}_{q-r-1} \\ &= \mathbf{I}_N + (\mathbf{I}_N - c_q\mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1}c_q(\mathbf{I}_{n+1} \otimes \mathbf{J}_w) \\ &= (\mathbf{I}_N - c_q\mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1}\end{aligned}$$

as desired, so $\mathbf{D}_{r+1}$ has the claimed form.

Finally, given $u', v' \in \{0, \ldots, 2^{r+1} - 1\}$ in the index set of $\mathbf{D}_{r+1}$, the algorithm can determine which indices u, v corresponding to rows/columns of $\mathbf{D}_r$ were used to produce $(\mathbf{D}_{r+1})_{u',v'}$. Given this, the machine can determine if $\mathbf{W}_{q-r-1}$ were left and/or right concatenated in $(\mathbf{D}_{r+1})_{u',v'}$ and recurse on u, v (and storing if each concatenation occurred requires only two bits per level). Since this requires storing a constant number of bits per level and the current level r, we require space $O(\log n)$ to output the index set. Furthermore, note that this algorithm does not depend on the branching program $\mathbf{B}$, only on n and the definition of $\mathbf{C}_q$. ◻

For the remainder of the section we index all relevant block matrices with respect to the conventional ordering of the cycle $\mathbf{C}_q$, dropping the π_q notation.

Corollary 5.20. *We have*

$$(\mathbf{L}^{-1})_{0,n} = (\mathbf{L}^{(0)})^{-1}_{0,n} = (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^n \mathbf{W}_0^n.$$

Proof. If we replace $\mathbf{W}_i$ with $\mathbf{W}_0^{2^i}$ in the construction above, by Equation 10 of [2] we have that $\mathbf{L}^{(i)} = \mathbf{L}$ for all i, so from Lemmas 5.19 and 5.18 we obtain that $\mathbf{L}^{-1}$ has the desired form. □

With this corollary, we obtain that an accurate estimate of $\mathbf{L}^{-1}$ implies an accurate estimate of $\mathbf{L}^{-1}_{0,n} = (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^n \mathbf{W}_0^n$, and by using Corollary 5.6 we will use this to bound the distance of the WPRG to $\mathbf{W}_0^n$.

5.5 Applying Richardson iteration

From the previous subsection, we obtain a matrix $(\mathbf{L}^{(q)})^{-1}$ which we will show is a good (enough) approximation of $\mathbf{L}^{-1}$ to apply preconditioned Richardson iterations (Proposition 2.2) to obtain a very accurate estimate of $\mathbf{L}^{-1}$.

Proposition 5.21. *Given $n \in \mathbb{N}$ and $\delta, \varepsilon \in (0, 1/2)$, let $\ell = \lceil \log_{1/\delta}(1/\varepsilon) \rceil$. Then for every length-$n$ permutation branching program $\mathbf{B}$, let $\mathbf{L}$ be defined as in Definition 5.16 and $\mathbf{L}^{(q)}$ as in Equation (5.1). Define $\mathbf{Err} = \mathbf{I}_{2^q N} - (\mathbf{L}^{(q)})^{-1}\mathbf{L}$ and define*

$$\widetilde{\mathbf{L}^{-1}} = \sum_{i=0}^{\ell} \mathbf{Err}^i \cdot (\mathbf{L}^{(q)})^{-1}.$$

Then $\|\widetilde{\mathbf{L}^{-1}} - \mathbf{L}^{-1}\|_{\max} \le \varepsilon \cdot \mathrm{poly}(n)$.

To prove this, we follow the argument of Ahmadinejad et al. [2] which in turn is based on Cohen et al. [19]. We wish to apply preconditioned Richardson iteration (Lemma 2.2), but to do so, we must first define an appropriate submultiplicative norm.

- For a matrix $\mathbf{A} \in \mathbb{C}^{d \times d}$ we write $\mathbf{A}^*$ to denote its conjugate transpose and write $\mathbf{U}_\mathbf{A} = (\mathbf{A} + \mathbf{A}^*)/2$ to denote its **symmetrization**.
- We say a Hermitian matrix $\mathbf{A}$ is **positive semidefinite** (PSD) or write $\mathbf{A} \succeq 0$ if $x\mathbf{A}x^* \ge 0$ for all $x \in \mathbb{C}^d$. For two Hermitian matrices $\mathbf{A}, \widetilde{\mathbf{A}}$, we use $\mathbf{A} \succeq \widetilde{\mathbf{A}}$ to denote $\mathbf{A} - \widetilde{\mathbf{A}} \succeq 0$ and define $\preceq$ analogously.
- For a PSD matrix $\mathbf{H}$, we define the seminorm induced by $\mathbf{H}$ as $\|x\|_\mathbf{H} = \sqrt{x\mathbf{H}x^*}$ and the corresponding matrix seminorm as $\|\mathbf{A}\|_\mathbf{H} = \max_{x \neq 0} \|x\mathbf{A}\|_\mathbf{H} / \|x\|_\mathbf{H}$. Note that $\|\cdot\|_\mathbf{H}$ is submultiplicative if $\mathbf{H}$ is invertible.

We give a basic proposition that allows us to pass from $\mathbf{H}$-norm to 2-norm.

Proposition 5.22. *Let $\mathbf{H} \in \mathbb{C}^{d\times d}$ be a Hermitian positive definite matrix with minimum eigenvalue α and maximum eigenvalue β. Then for every $x \in \mathbb{C}^d$, $\alpha\|x\|_2^2 \leq \|x\|_{\mathbf{H}}^2 \leq \beta\|x\|_2^2$.*

Proof. Let $(\mu_i)_{i\in[d]}$ be an orthonormal eigenbasis for $\mathbf{H}$ and $(\lambda_i)_{i\in[d]} \in \mathbb{R}$ the associated eigenvalues. Then for every $x \in \mathbb{C}^d$,

$$\|x\|_{\mathbf{H}}^2 = x\left(\sum_{i=1}^d \lambda_i \mu_i^* \mu_i\right) x^* = \sum_{i=1}^d \lambda_i |x\mu_i^*|^2 \leq \beta \sum_{i=1}^d |x\mu_i^*|^2 = \beta\|x\|_2^2$$

and the lower bound is nearly identical. □

Corollary 5.23. *Let $\mathbf{H} \in \mathbb{C}^{d\times d}$ be a Hermitian positive definite matrix with minimum eigenvalue α and maximum eigenvalue β. Then for every $\mathbf{A} \in \mathbb{C}^{d\times d}$, $\sqrt{\alpha/\beta}\|\mathbf{A}\|_2 \leq \|\mathbf{A}\|_{\mathbf{H}} \leq \sqrt{\beta/\alpha}\|\mathbf{A}\|_2$.*

Proof. Applying the previous proposition twice we have

$$\|\mathbf{A}\|_{\mathbf{H}} = \max_{x\neq 0} \frac{\|x\mathbf{A}\|_{\mathbf{H}}}{\|x\|_{\mathbf{H}}} \leq \sqrt{\beta/\alpha} \max_{x\neq 0} \frac{\|x\mathbf{A}\|_2}{\|x\|_2} = \sqrt{\beta/\alpha}\|\mathbf{A}\|_2$$

and the lower bound is nearly identical. □

Using these tools, we can build a positive definite matrix $\mathbf{F}$ such that $\mathbf{L}^{(q)}$ approximates $\mathbf{L}^{-1}$ with respect to $\|\cdot\|_{\mathbf{F}}$. For all $i \in \{0, \dots, q\}$, define

$$\mathbf{S}^{(i)} = \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I}_{2^{q-i}N} - \mathbf{C}_{q-i} \otimes c_i \mathbf{W}_i \end{bmatrix} \in \mathbb{R}^{2^q N \times 2^q N} \tag{5.3}$$

where the 0 padding is added to make the dimensions of the matrices equal, and recall

$$\mathbf{U}_{\mathbf{S}^{(i)}} = \frac{1}{2}\left(\mathbf{S}^{(i)} + (\mathbf{S}^{(i)})^T\right).$$

Lemma 5.24. *Let $\mathbf{S}^{(i)}$ and $\mathbf{L}^{(i)}$ be defined as in Equation (5.3) and Equation (5.1), respectively. Then defining $\mathbf{F} = \frac{2}{q}\sum_{i=0}^q \mathbf{U}_{\mathbf{S}^{(i)}}$, we have*

$$\left\|\mathbf{I}_{2^q N} - \widetilde{\mathbf{L}^{-1}}\mathbf{L}^{-1}\right\|_{\mathbf{F}} \leq \delta.$$

We follow the proof of [2], except that our choice of c makes all of our matrices $\mathbf{L}^{(i)}$ strictly diagonally dominant. The only aspect of the proof which differs is the observation that Schur complements of strictly diagonally dominant matrices are strictly diagonally dominant. As such, we defer the proof to Section 6.

With our choice of c, we obtain a bound of $1/\mathrm{poly}(n)$ on the minimum eigenvalue of $\mathbf{F}$.

Lemma 5.25. *Let $\mathbf{S}^{(i)}$ and $\mathbf{L}^{(i)}$ be defined as in Equation (5.3) and Equation (5.1), respectively, and define $\mathbf{F} = \frac{2}{q}\sum_{i=0}^q \mathbf{U}_{\mathbf{S}^{(i)}}$. The eigenvalues of $\mathbf{F}$ are contained in $[1/q(n+1), 3]$, and $\|\mathbf{L}^{-1}\|_2 \leq n+1$.*

Proof. For every $v \in \mathbb{R}^{2^q N}$ with $\|v\|_2 = 1$ we have

$$\|v\mathbf{L}\|_2 = \|v(\mathbf{I}_{2^q N} - \mathbf{C}_q \otimes c\mathbf{W}_0)\|_2 \in [1-c, 1+c] \subset [1/(n+1), 2]$$

which suffices to bound $\|\mathbf{L}^{-1}\|_2$ as desired. Furthermore, we have

$$\mathbf{U}_{\mathbf{S}^{(0)}} = \mathbf{I}_{2^q N} - \mathbf{U}_{\mathbf{C}_q \otimes c_0 \mathbf{W}_0}$$

and for all $i > 0$ we have

$$\mathbf{U}_{\mathbf{S}^{(i)}} = \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I}_{2^{q-i}N} - \mathbf{U}_{\mathbf{C}_{q-i} \otimes c_i \mathbf{W}_i} \end{bmatrix}$$

and since $\mathbf{U}_{\mathbf{C}_{q-i} \otimes \mathbf{W}_i}$ is a stochastic matrix, all eigenvalues of $\mathbf{U}_{\mathbf{S}^{(i)}}$ are contained in $[0, 2]$ for $i > 0$ and $[1-c, 2]$ for $i = 0$. Therefore for every $v \in \mathbb{R}^{2^q N}$ we have

$$\|v\mathbf{F}\|_2 = \left\| \frac{2}{q} \sum_{i=0}^{q} v\mathbf{U}_{\mathbf{S}^{(i)}} \right\|_2 \in [1/q(n+1), 3]. \qquad \square$$

Now that we have this $\mathbf{F}$ norm, we can use Richardson iteration to prove Proposition 5.21.

Proof. Applying Lemma 2.2 with $\mathbf{A} = \mathbf{L}$, $\mathbf{P}_0 = (\mathbf{L}^{(q)})^{-1}$, $m = \ell$, norm $\|\cdot\| = \|\cdot\|_{\mathbf{F}}$ and $\alpha = \delta$, we obtain

$$\left\|\mathbf{I}_{2^q N} - \widetilde{\mathbf{L}^{-1}}\mathbf{L}\right\|_{\mathbf{F}} = \left\|\mathbf{I}_{2^q N} - \left(\sum_{i=0}^{\ell} \mathbf{Err}^i \cdot (\mathbf{L}^{(q)})^{-1}\right)\mathbf{L}\right\|_{\mathbf{F}} = \|\mathbf{I}_{2^q N} - \mathbf{P}_m \mathbf{L}\|_{\mathbf{F}} \le \delta^{\ell} \le \varepsilon.$$

Thus,

$$\begin{aligned}
\left\|\mathbf{L}^{-1} - \widetilde{\mathbf{L}^{-1}}\right\|_{\max} &\le \left\|\mathbf{L}^{-1} - \widetilde{\mathbf{L}^{-1}}\right\|_2 \\
&= \left\|(\mathbf{L}^{-1} - \widetilde{\mathbf{L}^{-1}})\mathbf{L}\mathbf{L}^{-1}\right\|_2 \\
&\le \left\|\mathbf{I}_{2^q N} - \widetilde{\mathbf{L}^{-1}}\mathbf{L}\right\|_2 \cdot \|\mathbf{L}^{-1}\|_2 \\
&\le \left(\left\|\mathbf{I}_{2^q N} - \widetilde{\mathbf{L}^{-1}}\mathbf{L}\right\|_{\mathbf{F}} \cdot \sqrt{3q(n+1)}\right) \|\mathbf{L}^{-1}\|_2 && \text{(Proposition 5.23)} \\
&\le \varepsilon \cdot \mathrm{poly}(n) && \text{(Lemma 5.25)} \qquad \square
\end{aligned}$$

5.6 Interpretation as a weighted PRG

Now that we have a polynomial in the $\mathbf{W}$ approximating $\mathbf{L}^{-1}$ and thus $(\mathbf{L}^{-1})_{0,n} = (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^n \mathbf{W}_0^n$, to complete the proof of Theorem 5.2 we will interpret the polynomial as $(\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^n \mathbf{B}[\mathrm{GEN}_0]$ for a WPRG GEN_0.

Lemma 5.26. *Given $n \in \mathbb{N}$ and $\delta \in (0, 1/2)$, let INW_j be as defined in Definition 5.12 with the same n and δ. Then for all $x, y \in \{0, \dots, n\}$ there is a product of PRGs*

$$G_{x,y} = \mathrm{INW}_{i_1} \cdots \mathrm{INW}_{i_r}$$

where $r \leq 2q$ and given x, y the index set $i_1, \dots, i_r$ is computable in space $O(\log n)$. Furthermore, let $m = y - x \mod n + 1$. Then for every length-n permutation branching program $\mathbf{B}$ with lift transition matrix $\mathbf{B}$, let $\mathbf{L}^{(q)}$ be as defined in Equation (5.1). Then,

$$(\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^m \overline{\mathbf{B}^{(m)}} \left[G_{x,y}\right] = (\mathbf{L}^{(q)})^{-1}_{x,y}.$$

Proof. We apply the previous lemmas on the structure of $(\mathbf{L}^{(q)})^{-1}$.

$$\begin{aligned}
(\mathbf{L}^{(q)})^{-1}_{x,y} &= (\mathbf{D}_0)_{x,y} && \text{(Lemma 5.18)}\\
&= (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^m \mathbf{W}_{i_1} \cdots \mathbf{W}_{i_r} && \text{(Lemma 5.19)}\\
&= (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^m \overline{\mathbf{B}^{(2^{i_1})}} [\mathrm{INW}_{i_1}] \cdots \overline{\mathbf{B}^{(2^{i_r})}} [\mathrm{INW}_{i_r}] && \text{(Definition 5.12)}\\
&= (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^m \overline{\mathbf{B}^{(m)}} [\mathrm{INW}_{i_1} \cdots \mathrm{INW}_{i_r}]
\end{aligned}$$

where $r \leq 2q$ and given x, y the index set $i_1, \dots, i_r$ is computable in space $O(\log n)$. Furthermore, as the indices $i_1, \dots, i_r$ are independent of $\mathbf{B}$, this holds for all such branching programs $\mathbf{B}$ and we conclude. □

We can then describe the structure of $\mathbf{Err}$ in terms of combinations of these base PRGs.

Lemma 5.27. *For every $n \in \mathbb{N}$ and $\delta \in (0, 1/2)$ and $i, j \in \{0, \dots, n\}$, let $m = j - i \mod n + 1$. Let $\{G_{x,y} : x, y \in \{0, \dots, n\}\}$ be defined as in Lemma 5.26 with the same n and δ and recall R is the trivial PRG on one bit. Then for every length-n permutation branching program $\mathbf{B}$, let $\mathbf{Err} \in \mathbb{R}^{(n+1)N \times (n+1)N}$ be the $(n+1) \times (n+1)$ block matrix as defined in Proposition 5.21. Then,*

$$\mathbf{Err}_{i,j} = \begin{cases} 0 & i = j \\ c^m \overline{\mathbf{B}^{(m)}} \left[G_{i,j-1} R - G_{i,j}\right] & i \neq j. \end{cases}$$

Proof. The proof is nearly identical to that of Lemma 4.7. We detail the case where $i < j$.

$$\begin{aligned}
\mathbf{Err}_{i,j} &= -\sum_{k=0}^{n} (\mathbf{L}^{(q)})^{-1}_{i,k} \cdot \mathbf{L}_{k,j}\\
&= -\left[(\mathbf{L}^{(q)})^{-1}_{i,j} \cdot \mathbf{L}_{j,j} + (\mathbf{L}^{(q)})^{-1}_{i,j-1} \cdot \mathbf{L}_{j-1,j}\right] && \text{(Definition 5.16)}\\
&= -\left[(\mathbf{L}^{(q)})^{-1}_{i,j} \cdot \mathbf{I}_N + (\mathbf{L}^{(q)})^{-1}_{i,j-1} \cdot -c\mathbf{W}_0\right]\\
&= (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} \left(-c^{j-i} \overline{\mathbf{B}^{(m)}} \left[G_{i,j}\right] + c^{j-i-1} \overline{\mathbf{B}^{(m-1)}} \left[G_{i,j-1}\right] c\mathbf{W}_0\right) && \text{(Lemma 5.26)}\\
&= (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^{j-i} \left(\overline{\mathbf{B}^{(m)}} \left[G_{i,j-1} R\right] - \overline{\mathbf{B}^{(m)}} \left[G_{i,j}\right]\right)\\
&= c^{j-i} \overline{\mathbf{B}^{(m)}} \left[G_{i,j-1} R - G_{i,j}\right] && \text{(Corollary 5.6).}
\end{aligned}$$

and the case $i > j$ is nearly identical. Then for $i = j$,

$$\begin{aligned}
\mathbf{Err}_{i,i} &= \mathbf{I}_N - \sum_{k=0}^{n} (\mathbf{L}^{(q)})^{-1}_{i,k} \cdot \mathbf{L}_{k,j} \\
&= \mathbf{I}_N - \left[(\mathbf{L}^{(q)})^{-1}_{i,i} \cdot \mathbf{I}_N + (\mathbf{L}^{(q)})^{-1}_{i,i-1} \cdot -c\mathbf{W}_0 \right] && \text{(Definition 5.16)} \\
&= \mathbf{I}_N - (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} \left[\mathbf{I}_N - c^{n+1} \overline{\mathbf{B}^{(n+1)}} [G_{i,i-1} R] \right] && \text{(Lemma 5.26)} \\
&= \mathbf{I}_N - (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} \left[\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w \right] && \text{(Corollary 5.5).} \\
&= 0
\end{aligned}$$

□

We require one more lemma, which ensures that terms corresponding to WPRGs of length greater than n fall out of the error-reduction procedure.

Lemma 5.28. *For every length-n permutation branching program $\mathbf{B}$, let $\mathbf{Err} \in \mathbb{R}^{(n+1)N \times (n+1)N}$ be the $(n+1) \times (n+1)$ block matrix as defined in Proposition 5.21. For every sequence $0 = i_0, \ldots, i_r$ of indices where there exists j such that $i_j \geq i_{j+1}$,*

$$\prod_{j=1}^{r} \mathbf{Err}_{i_{j-1}, i_j} = 0.$$

Proof. If $i_j = i_{j+1}$ for any j the claim is trivially satisfied, so we assume adjacent indices are distinct. For $j \in [r]$ let $x_j = i_j - i_{j-1} \mod n+1$ and $x = \sum_{j=1}^{r} x_j$. By assumption, $x \geq n+1$. Then

$$\begin{aligned}
\prod_{j=1}^{r} \mathbf{Err}_{i_{j-1}, i_j} &= \prod_{j=1}^{r} c^{x_j} \overline{\mathbf{B}^{(x_j)}} [G_{i_{r-1}, i_r - 1} R - G_{i_{r-1}, i_r - 1}] && \text{(Lemma 5.27)} \\
&= c^x \overline{\mathbf{B}^{(x)}} \left[\prod_{j=1}^{r} (G_{i_{r-1}, i_r - 1} R - G_{i_{r-1}, i_r - 1}) \right] \\
&= 0
\end{aligned}$$

where the final line follows from writing the preceding line as a sum of 2^r PRGs with positive and negative signs and applying Corollary 5.6. □

We are now ready to describe the entries of $\widetilde{\mathbf{L}^{-1}}$. We define the index set in an analogous way to Definition 4.10.

Definition 5.29. For all $n, \ell \in \mathbb{N}$ define the index set $\mathbb{V}_{n,\ell}$ as

$$\mathbb{V}_{n,\ell} = \{0 = \sigma_0 < \sigma_1 < \cdots < \sigma_r \leq n : \sigma_i \in \mathbb{Z}^+, \quad 0 \leq r \leq \ell\}.$$

For $\sigma = (\sigma_0, \sigma_1, \ldots, \sigma_r) \in \mathbb{V}_{n,\ell}$ we write $|\sigma| = r$. Note that we include the empty tuple (0).

We now index the nonzero summands of the Richardson polynomial, equivalent to Lemma 4.11 with the products of INW PRGs G taking the place of NIS.

Lemma 5.30. *For all $n, \ell \in \mathbb{N}$ and $\delta \in (0, 1/2)$, let $\{G_{x,y} : x, y \in \{0, \ldots, n\}\}$ be defined as in Lemma 5.26 with the same n and δ and $\mathbb{V}_{n,\ell}$ as in Definition 5.29 with the same n, ℓ. Then for all $\sigma = (\sigma_0, \sigma_1, \ldots, \sigma_r) \in \mathbb{V}_{n,\ell}$, define the WPRG (using the sum and product rules of Definitions 3.6 and 3.9)*

$$M_\sigma = \prod_{i=0}^{r-1} (G_{\sigma_i, \sigma_{i+1}-1} R - G_{\sigma_i, \sigma_{i+1}}) G_{\sigma_r, n}$$

where $M_{(0)} = G_{0,n}$. For every length-n permutation branching program $\mathbf{B}$, let $\mathbf{Err}$ and $(\mathbf{L}^{(q)})^{-1}$ be defined as in Proposition 5.21. Then,

$$\left(\sum_{r=0}^{\ell} \mathbf{Err}^r \cdot (\mathbf{L}^{(q)})^{-1} \right)_{0,n} = (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^n \sum_{\sigma \in \mathbb{V}_{n,\ell}} \mathbf{B}^{(n)}[M_\sigma].$$

Proof. Fix any $r \in [\ell]$. Then we have

$$\begin{aligned}
&(\mathbf{Err}^r \cdot (\mathbf{L}^{(q)})^{-1})_{0,n} \\
&\quad = \sum_{(i_j) \in \{0..n\}^r} \mathbf{Err}_{0,i_1} \left(\prod_{j=1}^{r-1} \mathbf{Err}_{i_j, i_{j+1}} \right) (\mathbf{L}^{(q)})^{-1}_{i_r, n} \\
&\quad = \sum_{\sigma \in \mathbb{V}_{n,\ell} : |\sigma| = r} \left(\prod_{i=0}^{r-1} \mathbf{Err}_{\sigma_i, \sigma_{i+1}} \right) (\mathbf{L}^{(q)})^{-1}_{\sigma_r, n} && \text{(Lemma 5.28)} \\
&\quad = \sum_{\sigma \in \mathbb{V}_{n,\ell} : |\sigma| = r} \prod_{i=0}^{r-1} c^{\sigma_{i+1} - \sigma_i} \overline{\mathbf{B}^{(\sigma_{i+1} - \sigma_i)}} \left[G_{\sigma_i, \sigma_{i+1}-1} R - G_{\sigma_i, \sigma_{i+1}} \right] (\mathbf{L}^{(q)})^{-1}_{\sigma_r, n} && \text{(Lemma 5.27)} \\
&\quad = (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^n \sum_{\sigma \in \mathbb{V}_{n,\ell} : |\sigma| = r} \overline{\mathbf{B}^{(n)}} \left[M_\sigma \right] && \text{(Lemma 5.26).}
\end{aligned}$$

Then for $r = 0$ we have

$$(\mathbf{L}^{(q)})^{-1}_{0,n} = (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^n \overline{\mathbf{B}^{(n)}} \left[G_{0,n} \right] = (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^n \overline{\mathbf{B}^{(n)}} \left[M_{(0)} \right]$$

so we conclude. $\square$

We then prove an analogue of Corollary 4.12.

Corollary 5.31. *Given $n, \ell \in \mathbb{N}$ and $\delta \in (0, 1/2)$, let $\mathbb{V}_{n,\ell}$ be defined as in Definition 5.29 with the same n, ℓ and $\{M_\sigma : \sigma \in \mathbb{V}_{n,\ell}\}$ as in Lemma 5.30 with the same n, δ. For all $\sigma \in \mathbb{V}_{n,\ell}$, let $r = |\sigma|$. Then*

$$M_\sigma = \sum_{x \in \{0,1\}^r} \tau_{\sigma,x} \cdot P_{\sigma,x,1} \cdots P_{\sigma,x,k},$$

where $k \le 3q(r+1)$ and for all σ, x, i we have that $\tau_{\sigma,x} \in \{-1,1\}$ and $P_{\sigma,x,i}$ is an explicit PRG with seed length $s = O(\log n \cdot \log(\log(n)/\delta))$. Furthermore given $\sigma \in \mathbb{V}_{n,\ell}$, $x \in \{0,1\}^r$ and $i \in [k]$, $\tau_{\sigma,x}$ can be computed and $P_{\sigma,x,i}$ can be evaluated in space $O(s + \log|\mathbb{V}_{n,\ell}|)$.

Proof. For every $\sigma \in \mathbb{V}_{n,\ell}$ and $x \in \{0,1\}^r$, let $\tau_{\sigma,x} = (-1)^{\sum_{i=1}^r x_i}$. For all $i \in [r]$, define

$$D_{\sigma,x,i} = \begin{cases} G_{\sigma_{i-1},\sigma_i - 1} R & x_i = 0 \\ G_{\sigma_{i-1},\sigma_i} & x_i = 1 \end{cases}$$

and define $D_{\sigma,x,r+1} = G_{\sigma_r,n}$, where by Lemma 5.26 each $D_{\sigma,x}$ is a product of at most $3q$ explicit PRGs, each with seed length $s = O(\log n \cdot \log(\log(n)/\delta))$, and given $\sigma \in \mathbb{V}_{n,\ell}$, $x \in \{0,1\}^r$ and $i \in [r]$ the index set of the PRGs in $D_{\sigma,x,i}$ can be computed in space $O(\log n)$. Then define

$$\prod_{j=1}^{k} P_{\sigma,x,j} = \prod_{i=1}^{r+1} D_{\sigma,x,i},$$

where given $\sigma \in \mathbb{V}_{n,\ell}$ and $x \in \{0,1\}^r$, every index $j \in [k]$ corresponds to a base PRG in the right hand product (and this PRG is computable in the desired space bound by Lemma 5.26), and $k \le (r+1)3q$. Finally, by definition

$$M_\sigma = \sum_{x \in \{0,1\}^r} \tau_{\sigma,x} \cdot P_{\sigma,x,1} \cdots P_{\sigma,x,k}. \qquad \square$$

We next show that the family of WPRGs jointly approximate $\overline{\mathbf{B}^{(n)}}[U_n]$, which holds the distribution of random walks from layer 0 to layer n in the branching program.

Lemma 5.32. *Given $n \in \mathbb{N}$ and $\varepsilon, \delta \in (0, 1/2)$, let $\ell = \lceil \log_{1/\delta}(1/\varepsilon) \rceil$ and let $\{M_\sigma : \sigma \in \mathbb{V}_{n,\ell}\}$ be defined as in Lemma 5.30 with the same n, ℓ and δ. Then for every length-n permutation branching program $\mathbf{B}$ with lift transition matrix $\mathbf{B}$,*

$$\left\| \sum_{\sigma \in \mathbb{V}_{n,\ell}} \overline{\mathbf{B}^{(n)}}[M_\sigma] - \overline{\mathbf{B}^{(n)}}[U_n] \right\|_{\max} \le \varepsilon \cdot \mathrm{poly}(n).$$

Proof. Let $\widetilde{\mathbf{L}^{-1}}$ and $\mathbf{L}$ be defined as in Proposition 5.21. We obtain

$$\begin{aligned}
&\varepsilon \cdot \mathrm{poly}(n) \\
&\quad \geq \left\| (\widetilde{\mathbf{L}^{-1}} - \mathbf{L}^{-1})_{0,n} \right\|_{\max} && \text{(Prop. 5.21)} \\
&\quad = \left\| (\widetilde{\mathbf{L}^{-1}})_{0,n} - (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^n \mathbf{W}_0^n \right\|_{\max} && \text{(Cor. 5.20)} \\
&\quad = \left\| (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^n \sum_{\sigma \in \mathbb{V}_{n,\ell}} \overline{\mathbf{B}^{(n)}}\,[M_\sigma] - (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^n \mathbf{W}_0^n \right\|_{\max} && \text{(Lemma 5.30)} \\
&\quad = \left\| c^n \sum_{\sigma \in \mathbb{V}_{n,\ell}} \overline{\mathbf{B}^{(n)}}\,[M_\sigma] - c^n \overline{\mathbf{B}^{(n)}}\,[U_n] \right\|_{\max} && \text{(Cor. 5.6)} \\
&\quad \geq \frac{1}{4} \left\| \sum_{\sigma \in \mathbb{V}_{n,\ell}} \overline{\mathbf{B}^{(n)}}\,[M_\sigma] - \overline{\mathbf{B}^{(n)}}\,[U_n] \right\|_{\max}
\end{aligned}$$

where the final inequality comes from our choice of $c = 1 - 1/(n+1)$, and the third equality follows from writing

$$\sum_{\sigma \in \mathbb{V}_{n,\ell}} M_\sigma = \sum_{\sigma \in \mathbb{V}_{n,\ell} \setminus (0)} M_\sigma + M_{(0)}$$

and noting that $(\mathbf{I}_{n+1} \otimes \mathbf{J}_w)\overline{\mathbf{B}^{(n)}}\left[\sum_{\sigma \in \mathbb{V}_{n,\ell} \setminus (0)} M_\sigma\right] = 0$ by Corollary 5.6 (as all M_σ except $M_{(0)}$ are a sum over $2^{|\sigma|}$ PRGs with positive and negative sign by Corollary 5.31) and $(\mathbf{I}_{n+1} \otimes \mathbf{J}_w)\overline{\mathbf{B}^{(n)}}\left[M_{(0)}\right] = (\mathbf{I}_{n+1} \otimes \mathbf{J}_w)\mathbf{W}_0^n$ by Corollary 5.6. ☐

We remark that the exact cancellation of all terms that do not correspond to WPRG outputs of length n is the motivation for our placement of $\mathbf{J}_w$ in the lift transition matrix, rather than some arbitrary transition function as in [26]. We are now prepared to prove Theorem 5.2.

Theorem 5.2. *Given $n \in \mathbb{N}$ and $\varepsilon, \delta \in (0, 1/2)$, let $\ell = O(\log_{1/\delta}(n/\varepsilon))$. Then there exists an explicit weighted generator GEN_0 such that GEN_0 is ε-pseudorandom for the class of permutation branching programs of length n and arbitrary width with respect to $\|\cdot\|_{\max}$ and*

$$\mathrm{GEN}_0 = \sum_{i \in [V]} \tau_i \cdot P_{i,1} P_{i,2} \cdots P_{i,k}$$

such that

1. *$V = n^{O(\ell)}$*
2. *$k = O(\ell \cdot \log n)$*
3. *For all i, $\tau_i \in \{-1, 1\}$.*

4. *For all i, j, $P_{i,j}$ is an (unweighted) PRG with seed length $s = O(\log n \cdot \log(\log(n)/\delta))$.*

5. *Given $i \in [V]$ and $j \in [k]$, τ_i and $P_{i,j}$ are evaluatable in space $O(s + \log V)$.*

Proof. Let $\{M_\sigma : \sigma \in \mathbb{V}_{n,\ell}\}$ be defined as in Lemma 5.30 with the same n, ℓ and δ, and let

$$\{\tau_i \cdot P_{i,1} \cdots P_{i,k} : \sigma \in \mathbb{V}_{n,\ell}, x \in \{0,1\}^{|\sigma|}\}$$

be the family obtained from Corollary 5.31 ranging over σ. Let $[V]$ be the set of terms (σ, x) and define

$$\mathrm{GEN}_0 = \sum_{i \in [V]} \tau_i \cdot P_{i,1} \cdots P_{i,k}.$$

All explicitness and seed length conditions are satisfied from Corollary 5.31, and we have $V = n^{O(\ell)}$ as desired. Finally, fixing an arbitrary length-n permutation branching program $\mathbf{B}$,

$$\begin{aligned}
\varepsilon \cdot \mathrm{poly}(n) &\geq \left\| \sum_{\sigma \in \mathbb{V}_{n,\ell}} \overline{\mathbf{B}^{(n)}}\,[M_\sigma] - \overline{\mathbf{B}^{(n)}}\,[U_n] \right\|_{\max} && \text{(Lemma 5.32)} \\
&= \left\| \sum_{i \in [V]} \overline{\mathbf{B}^{(n)}}\,[\tau_i \cdot P_{i,1} \cdots P_{i,k}] - \overline{\mathbf{B}^{(n)}}\,[U_n] \right\|_{\max} && \text{(Corollary 5.31)} \\
&\geq \left\| \sum_{i \in [V]} \left(\overline{\mathbf{B}^{(n)}}\,[\tau_i \cdot P_{i,1} \cdots P_{i,k}]\right)_{0,n} - \left(\overline{\mathbf{B}^{(n)}}\,[U_n]\right)_{0,n} \right\|_{\max} \\
&= \left\| \overline{\mathbf{B}} \left[\sum_{i \in [V]} \tau_i \cdot P_{i,1} \cdots P_{i,k} \right] - \overline{\mathbf{B}}[U_n] \right\|_{\max} && \text{(Proposition 5.4)} \\
&= \left\| \overline{\mathbf{B}}[\mathrm{GEN}_0] - \overline{\mathbf{B}}[U_n] \right\|_{\max}.
\end{aligned}$$

Finally, taking $\varepsilon \leftarrow \varepsilon/\mathrm{poly}(n)$ completes the proof. $\square$

5.7 Shorter seed length via derandomized PRG products

We now have a set of explicit WPRGs whose sum provides a high-quality approximation of an arbitrary permutation branching program of length n. As in Section 4, we wish to decrease the seed length of each summand. Applying Corollary 4.16 would give a nearly-logarithmic dependence on width. To obtain a seed length independent of width, we use the main result of [26].

Theorem 5.33 ([26] Theorem 1.4)**.** *For every $k, d \in \mathbb{N}$ and $\delta \in (0, 1/2)$, there is an explicit PRG $H : \{0,1\}^{s_{\mathrm{INW}}} \to [d]^k$ with seed length $s_{\mathrm{INW}} = O(\log(d) + \log(k)(\log(1/\delta) + \log\log(k))$ such that for every permutation branching program $\mathbf{B}$ of length k and degree d,*

$$\left\| \overline{\mathbf{B}}[H] - \overline{\mathbf{B}}[U_n] \right\|_{\max} \leq \delta.$$

We now state the inner derandomization lemma, the analogue of Corollary 4.16.

Lemma 5.34. *Fix $\gamma \in (0, 1/2)$ and a family of length-n WPRGs, $\{\tau_i \cdot P_{i,1} \cdots P_{i,k} : i \in [V]\}$, satisfying the following conditions.*

- *For all i, $\tau_i \in \{-1, 1\}$.*
- *For all i_j, $P_{i,j}$ is an explicit PRG with seed length s.*
- *Given i, j, the coefficient τ_i can be computed and the generator $P_{i,j}$ can be evaluated in space $O(s + \log V)$.*

Then there exists a $2V$-bounded explicit WPRG GEN with seed length $O(s + \log k \cdot \log(V \log(k)/\gamma))$ such that for every length-n permutation branching program $\mathbf{B}$,

$$\left\| \overline{\mathbf{B}}[\mathrm{GEN}] - \overline{\mathbf{B}}\left[\sum_{i \in [V]} \tau_i \cdot P_{i,1} \cdots P_{i,k}\right] \right\|_{\max} \leq \gamma.$$

Proof. Fix an arbitrary permutation branching program $\mathbf{B}$ of length n and width w (and degree 2). Let $H : \{0,1\}^{s_{\mathrm{INW}}} \to (\{0,1\}^s)^k$ be the PRG obtained from applying Theorem 5.33 with $k = k$, $d = 2^s$ and error $\delta = \gamma/V$.

Now fix arbitrary $i \in [V]$ and consider the product $P_{i,1} \cdots P_{i,k}$. For every $j \in [k]$ let $\{l_{j-1} + 1, l_{j-1} + 2, \ldots, l_j\}$ be the bits of the product output by $P_{i,j}$, where $l_0 = 0$, and define $\mathbf{B}'_{j-1..j}[s] = \mathbf{B}_{l_{j-1}..l_j}[P_{i,j}(s)]$. Note that this defines a length 1, degree 2^s permutation branching program of the same width as $\mathbf{B}$. Then $\mathbf{B}'$ is a degree 2^s, length k permutation branching program. Unrolling the definition,

$$\begin{aligned}
\gamma/V &\geq \left\| \overline{\mathbf{B}}'[U_{[2^s]^k}] - \overline{\mathbf{B}}'[H] \right\|_{\max} \\
&= \left\| \prod_{j=1}^{k} \mathbb{E}[\mathbf{B}_{l_{j-1}..l_j}[P_{i,j}(U_s)]] - \mathop{\mathbb{E}}_{x \leftarrow U_{s_{\mathrm{INW}}}} \left[\prod_{j=1}^{k} \mathbf{B}_{l_{j-1}..l_j}[P_{i,j}(H(x)_j)] \right] \right\|_{\max} \\
&= \left\| \overline{\mathbf{B}}\left[\prod_{j=1}^{k} P_{i,j}\right] - \overline{\mathbf{B}}\left[\left(\prod_{j=1}^{k} P_{i,j}\right) \circ H\right] \right\|_{\max}
\end{aligned}$$

where $H(x)_j$ is the jth symbol output by H on seed x. Then for all i, define

$$\mathrm{GEN}_i = \left(\prod_{j=1}^{k} P_{i,j}\right) \circ H$$

which is explicit by composition of space bounded algorithms and has seed length $s_{\mathrm{INW}} = O(s + \log(k)\log(V \log(k)/\gamma))$. Finally, we apply Proposition 3.8 and define the explicit WPRG

$$\mathrm{GEN} = \sum_{i \in [V]} \tau_i \cdot \mathrm{GEN}_i$$

which by the proposition is $2V$-bounded and has seed length

$$s_{\mathrm{INW}} + O(\log V) = O(s + \log k \cdot \log(V \log(k)/\gamma)).$$

Finally,

$$\begin{aligned}
\left\|\overline{\mathbf{B}}[\mathrm{GEN}] - \overline{\mathbf{B}}\left[\sum_{i\in[V]} \tau_i \cdot P_{i,1}\cdots P_{i,k}\right]\right\|_{\max} &= \left\|\overline{\mathbf{B}}\left[\sum_{i\in[V]} \tau_i \cdot \mathrm{GEN}_i\right] - \overline{\mathbf{B}}\left[\sum_{i\in[V]} \tau_i \cdot P_{i,1}\cdots P_{i,k}\right]\right\|_{\max} \\
&\le \sum_{i\in[V]} \left\|\overline{\mathbf{B}}[\mathrm{GEN}_i] - \overline{\mathbf{B}}[P_{i,1}\cdots P_{i,k}]\right\|_{\max} \\
&\le \frac{\gamma}{V}\cdot V. \qquad \square
\end{aligned}$$

5.8 Putting it all together

We are now prepared to prove our main theorem.

Theorem 5.1. *For all $n \in \mathbb{N}$ and $\varepsilon \in (0, 1/2)$, there exists an explicit ε-WPRG for the class of permutation branching programs of length n with respect to $\|\cdot\|_{\max}$ with seed length*

$$O(\log(n)\sqrt{\log(n/\varepsilon)}\sqrt{\log\log(n/\varepsilon)} + \log(1/\varepsilon)\log\log(n/\varepsilon)).$$

Proof. Applying Theorem 5.2 with $n = n$, $\varepsilon = \varepsilon/2$ and $\delta = \delta$ to be chosen later, we obtain

$$\ell = \lceil \log_{1/\delta}(n/\varepsilon)\rceil = O(\log(n/\varepsilon)/\log(1/\delta))$$

and a generator

$$\mathrm{GEN}_0 = \sum_{i\in[V]} \tau_i \cdot P_{i,1}P_{i,2}\cdots P_{i,k}$$

satisfying for every length-n permutation branching program $\mathbf{B}$,

$$\left\|\overline{\mathbf{B}}[\mathrm{GEN}_0] - \overline{\mathbf{B}}[U_n]\right\|_{\max} \le \varepsilon/2.$$

Furthermore, the family $\{\tau_i \cdot P_{i,1}P_{i,2}\cdots P_{i,k} : i \in [V]\}$ satisfies the requirements of Lemma 5.34 with $V \le |\mathbb{V}_{n,\ell}|2^{\ell} = n^{O(\ell)}$ and $k \le 5\log(n)\ell$ and $s = O(\log n \cdot \log(\log(n)/\delta))$. Therefore, let GEN be the WPRG obtained from applying Lemma 5.34 to this family with error $\gamma = \varepsilon/2$.

Then unwinding definitions, we obtain

$$\begin{aligned}
\left\|\overline{\mathbf{B}}[\mathrm{GEN}] - \overline{\mathbf{B}}[U_n]\right\|_{\max} &\le \left\|\overline{\mathbf{B}}[\mathrm{GEN}] - \overline{\mathbf{B}}[\mathrm{GEN}_0]\right\|_{\max} + \left\|\overline{\mathbf{B}}[\mathrm{GEN}_0] - \overline{\mathbf{B}}[U_n]\right\|_{\max} \\
&\le \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon
\end{aligned}$$

where the last line follows from our choice of error in Lemma 5.34 and Theorem 5.2.

It remains to optimize parameters. By Lemma 5.34, GEN is explicit and has seed length

$$s = O\left(\log(n)\log(\log(n)/\delta) + \log(\log(n)\ell)(\ell + \log(|\mathbb{V}_{n,\ell}|/\varepsilon))\right).$$

Finally, we choose $\delta = 2^{-\sqrt{\log(n/\varepsilon)\log\log(n/\varepsilon)}}$. Thus we obtain

$$\ell = O\left(\sqrt{\log(n/\varepsilon)}/\sqrt{\log\log(n/\varepsilon)}\right)$$

which implies $\log|\mathbb{V}_{n,\ell}| = O(\ell\log(n)) = O\left(\log(n)\sqrt{\log(n/\varepsilon)}/\sqrt{\log\log(n/\varepsilon)}\right)$, which implies a final seed length of

$$\begin{aligned}
s &= O\left(\log(n)\log\log(n) + \log(n)\sqrt{\log(n/\varepsilon)\log\log(n/\varepsilon)} + \log\log(n/\varepsilon)(\ell + \log(|\mathbb{V}_{n,\ell}|/\varepsilon))\right)\\
&= O\left(\log(n)\sqrt{\log(n/\varepsilon)\log\log(n/\varepsilon)} + \frac{\log\log(n/\varepsilon)}{\sqrt{\log\log(n/\varepsilon)}}\sqrt{\log(n/\varepsilon)}\log(n) + \log\log(n/\varepsilon)\log(1/\varepsilon)\right)\\
&= O\left(\log(n)\sqrt{\log(n/\varepsilon)}\sqrt{\log\log(n/\varepsilon)} + \log(1/\varepsilon)\log\log(n/\varepsilon)\right). \qquad \square
\end{aligned}$$

6 Proof of Lemma 5.24

We require two prior results. We first recall notation.

- For a matrix $\mathbf{A}$, we use $\mathbf{A}^+$ to denote the (Moore–Penrose) pseudoinverse of $\mathbf{A}$.
- For a PSD matrix $\mathbf{X}$, we let $\mathbf{X}^{1/2}$ to denote the square root of $\mathbf{X}$, which is the unique PSD matrix such that $\mathbf{X}^{1/2}\mathbf{X}^{1/2} = \mathbf{X}$. Furthermore, let $\mathbf{X}^{+/2}$ denote the pseudoinverse of the square root of $\mathbf{X}$.

In our case we exclusively work with invertible matrices, so $\mathbf{A}^+ = \mathbf{A}^{-1}$, but we state the results in their original form.

Lemma 6.1 ([2, Lemma 6.7]). *Let $\mathbf{S}^{(0)}, \ldots, \mathbf{S}^{(q)}$ and $\mathbf{L}^{(0)}, \ldots, \mathbf{L}^{(q)}$ be defined as in Equations (5.3) and (5.1), respectively.*

Then for

$$\mathbf{F} = \frac{2}{q}\sum_{i=0}^{q}\mathbf{U}_{\mathbf{S}^{(i)}}$$

we have the following two properties.

- *For each $0 \le i \le q$,*
$$\left\|\mathbf{F}^{+/2}(\mathbf{L} - \mathbf{L}^{(i)})\mathbf{F}^{+/2}\right\|_2 \le \delta/40q.$$

- *The final matrix* $\mathbf{L}^{(q)}$ *satisfies*

$$\mathbf{L}^{(q)^T}\mathbf{F}^{+}\mathbf{L}^{(q)} \geq \frac{1}{40q^2}\mathbf{F}.$$

The lemma is proved in [2] for Laplacians of cycle lifts of Eulerian graphs without the scaling factor c. We reproduce the proof in Section 7, with a small modification to account for the strictly diagonally dominant $\mathbf{S}^{(i)}$. We now recall a further lemma required to bound the norm of **Err**.

Lemma 6.2 ([2, Lemma D.4])**.** *Suppose we are given matrices* $\mathbf{L}$, $\widetilde{\mathbf{L}}$ *and a positive semidefinite matrix* $\mathbf{F}$ *such that* $\ker(\mathbf{F}) \subseteq \ker(\mathbf{L}) = \ker(\mathbf{L}^T) = \ker(\widetilde{\mathbf{L}}) = \ker(\widetilde{\mathbf{L}}^T)$ *and*

- $\|\mathbf{F}^{+/2}(\mathbf{L} - \widetilde{\mathbf{L}})\mathbf{F}^{+/2}\|_2 \leq \delta$,
- $\widetilde{\mathbf{L}}\mathbf{F}^{+}\widetilde{\mathbf{L}} \geq \gamma\mathbf{F}$,

then $\|\mathbf{I}_{im(\mathbf{F})} - \widetilde{\mathbf{L}}^{+}\mathbf{L}\|_{\mathbf{F}} \leq \delta\sqrt{\gamma^{-1}}$.

Now we restate and prove the lemma.

Lemma 6.3. *Let* $\mathbf{S}^{(i)}$ *and* $\mathbf{L}^{(i)}$ *be defined as in Equation (5.3) and Equation (5.1), respectively. Then defining* $\mathbf{F} = \frac{2}{q}\sum_{i=0}^{q}\mathbf{U}_{\mathbf{S}^{(i)}}$, *we have*

$$\left\|\mathbf{I}_{2^qN} - \widetilde{\mathbf{L}^{-1}}\mathbf{L}^{-1}\right\|_{\mathbf{F}} \leq \delta.$$

Proof. We apply Lemma 6.2 with $\delta = \delta/40q$, $\mathbf{L} = \mathbf{L}, \widetilde{\mathbf{L}} = \mathbf{L}^{(q)}$ and $\mathbf{F} = \mathbf{F}$, all of which are invertible and thus immediately satisfy the kernel conditions, and satisfy the other conditions by Lemma 6.1, which gives

$$\|\mathbf{I}_{2^qN} - (\mathbf{L}^{(q)})^{-1}\mathbf{L}\|_{\mathbf{F}} \leq 40q\delta/40q = \delta.$$

□

7 Proof of Lemma 6.1

Here we reproduce the proof of [2, Lemma 6.7] to verify our claim. To maintain consistency with [2], we return to convention and index matrices starting from 1. First we recall the formal definition of a Schur complement.

Definition 7.1 (Schur complement)**.** For a matrix $\mathbf{A} \in \mathbb{C}^{N\times N}$ and $F, C \subseteq [N]$, let $\mathbf{A}_{FC}$ be the submatrix induced by the rows of F and columns of C. If F, C partition $[N]$ and $\mathbf{A}_{FF}$ is invertible, then we define the Schur complement of $\mathbf{A}$ onto the set C by

$$\mathrm{Sc}(\mathbf{A}, C) = \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{A}_{CC} - \mathbf{A}_{CF}\mathbf{A}_{FF}^{-1}\mathbf{A}_{FC} \end{bmatrix}.$$

Note that the standard definition is $\mathrm{Sc}(\mathbf{A}, C) = \mathbf{A}_{CC} - \mathbf{A}_{CF}\mathbf{A}_{FF}^{-1}\mathbf{A}_{FC}$, but we define the Schur complement as dimension-preserving by padding.

We then recall the lemma of Cohen et al. [18] that forms the core of the proof.

Lemma 7.2 ([18, Lemma 2.3]). *Consider a sequence $\mathbf{S}^{(0)}, \dots, \mathbf{S}^{(m)}$ of m-by-m matrices such that*

1. *$\mathbf{S}^{(i)}$ has nonzero indices only on the indices $[i+1, m]$,*
2. *the left-right kernels of $\mathbf{S}^{(i)}$ are equal, and after restricting $\mathbf{S}^{(i)}$ to the indices $[i+1, m]$, the kernel of the resulting matrix equals the coordinate restriction of the vectors in the kernel of $\mathbf{S}^{(0)}$, and*
3. *the symmetrization of each $\mathbf{S}^{(i)}$, denoted $\mathbf{U}_{\mathbf{S}^{(i)}}$, is positive semidefinite.*

Let $\mathbf{M} = \mathbf{M}^{(0)} = \mathbf{S}^{(0)}$ and define $\mathbf{M}^{(1)}, \dots, \mathbf{M}^{(m)}$ iteratively by

$$\mathbf{M}^{(i+1)} = \mathbf{M}^{(i)} + (\mathbf{S}^{(i+1)} - \mathrm{Sc}(\mathbf{M}^{(i)}, [i+1, m]))$$

If for a subsequence of indices $1 = i_0 < i_1 < \dots < i_{p_{\max}}$ associated scaling parameters $0 < \theta_0, \dots, \theta_{p_{\max}-1} < 1/2$ such that $\sum_{p=0}^{p_{\max}-1} \theta_p = 1$, and some global error $0 < \delta < 1/2$, we have for every $0 \le p < p_{\max}$

$$\|\mathbf{U}^{+/2}_{\mathbf{S}^{(i_p)}}(\mathbf{M}^{(i_p)} - \mathbf{M}^{(i_{p+1})})\mathbf{U}^{+/2}_{\mathbf{S}^{(i_p)}}\| \le \theta_p \delta$$

then for a matrix-norm defined from the symmetrization of the $\mathbf{S}^{(i_p)}$ matrices and the scaling parameters

$$\mathbf{F} = \sum_{0 \le p < p_{\max}} \theta_p \mathbf{U}_{\mathbf{S}^{(i_p)}}$$

we have the following.

1. *For each $0 \le i \le p_{\max}$,*
$$\|\mathbf{F}^{+/2}(\mathbf{M} - \mathbf{M}^{(i)})\mathbf{F}^{+/2}\|_2 \le \delta\,.$$
2. *The final matrix $\mathbf{M}^{(p_{\max})}$ satisfies*
$$\mathbf{M}^{(p_{\max})^T}\mathbf{F}^{+}\mathbf{M}^{(p_{\max})} \succeq \frac{1}{10 p_{\max}^2}\mathbf{F}.$$

We require one more basic derivation and two statements on unit circle equivalence.

Lemma 7.3 ([2, Lemma D.2]). *Let $\mathbf{L}^{(i)}$ be the $2^q N \times 2^q N$ matrices defined in Equation (5.1). Then*

$$\mathbf{L}^{(i+1)} - \mathbf{L}^{(i)} = \begin{bmatrix} 0 & 0 \\ 0 & -\mathbf{C}_{q-i-1} \otimes c_{i+1}\mathbf{W}_{i+1} + \mathbf{C}_{q-i-1} \otimes c_{i+1}\mathbf{W}_i^2 \end{bmatrix}.$$

This is proven without the scaling factor c_{i+1} but the construction is identical.

Lemma 7.4 ([2, Corollary 4.6]). *Let $\widetilde{\mathbf{W}}, \mathbf{W} \in \mathbb{C}^{N \times N}$ be (not necessarily symmetric) matrices such that $\widetilde{\mathbf{W}} \overset{\circ}{\approx}_\delta \mathbf{W}$. For all $k \in \mathbb{N}$ let $\mathbf{C}_{(k)}$ be the transition matrix for the directed cycle on k vertices. Then $\mathbf{C}_{(k)} \otimes \widetilde{\mathbf{W}} \overset{\circ}{\approx}_\delta \mathbf{C}_{(k)} \otimes \mathbf{W}$.*

Lemma 7.5 ([2, Lemma 3.8])**.** *Let* $\widetilde{\mathbf{W}}, \mathbf{W} \in \mathbb{C}^{N \times N}$ *be possibly asymmetric matrices. Then if* $\widetilde{\mathbf{W}} \overset{\circ}{\approx}_\delta \mathbf{W}$ *we have* $\left\| \mathbf{U}^{+/2}_{\mathbf{I}_N - \mathbf{W}} (\widetilde{\mathbf{W}} - \mathbf{W}) \mathbf{U}^{+/2}_{\mathbf{I}_N - \mathbf{W}} \right\|_2 \leq \delta$.

We are now prepared to prove the result.

Lemma 7.6 ([2, Lemma 6.7])**.** *Let* $\mathbf{S}^{(0)}, \ldots, \mathbf{S}^{(q)}$ *and* $\mathbf{L}^{(0)}, \ldots, \mathbf{L}^{(q)}$ *be defined as in Equations (5.3) and (5.1), respectively.*

Then for

$$\mathbf{F} = \frac{2}{q} \sum_{i=0}^{q} \mathbf{U}_{\mathbf{S}^{(i)}}$$

we have the following two properties.

- *For each* $0 \leq i \leq q$,

$$\left\| \mathbf{F}^{+/2} (\mathbf{L} - \mathbf{L}^{(i)}) \mathbf{F}^{+/2} \right\|_2 \leq \delta / 40q.$$

- *The final matrix* $\mathbf{L}^{(q)}$ *satisfies*

$$\mathbf{L}^{(q)^T} \mathbf{F}^{+} \mathbf{L}^{(q)} \geq \frac{1}{40q^2} \mathbf{F}.$$

Proof. From the $\mathbf{S}^{(i)}$ and the $\mathbf{L}^{(i)}$ we build a sequence of matrices $\hat{\mathbf{S}}^{(j)}$ and $\hat{\mathbf{M}}^{(j)}$ that satisfy the conditions of Lemma 7.2, and using that we derive the statement of the lemma. For $0 \leq i < q$, and $0 \leq j < 2^{q-i-1}N$, define $a_i = (2^q - 2^{q-i})N$, $\hat{\mathbf{S}}^{(a_q)} = \mathbf{S}^{(q)}$, and

$$\hat{\mathbf{S}}^{(a_i + j)} = \begin{cases} \mathbf{S}^{(i)} & \text{if } j = 0 \\ \mathrm{Sc}(\hat{\mathbf{M}}^{a_i + j - 1}, [a_i + j, 2^q N]) & \text{otherwise} \end{cases}$$

and

$$\hat{\mathbf{M}}^{(h+1)} = \hat{\mathbf{M}}^{(h)} + (\hat{\mathbf{S}}^{(h+1)} - \mathrm{Sc}(\hat{\mathbf{M}}^{(h)}, [h+1, 2^q N])) \quad \forall 0 \leq h \leq (2^q - 1)N$$

Note that the $\hat{\mathbf{S}}^{(i)}$ satisfy all the three premises in Lemma 7.2. First $\hat{\mathbf{S}}^{(i)}$ has non-zero entries only on the indices $[i+1, 2^q N]$. Furthermore, as $\hat{\mathbf{S}}^{(i)}$ restricted to the indices $[i+1, 2^q N]$ is a Schur complement of a diagonally dominant matrix and diagonal dominance is preserved under Schur complements, all the restricted $\hat{\mathbf{S}}^{(i)}$ are invertible so the kernel conditions are trivially satisfied and $\mathbf{U}_{\hat{\mathbf{S}}^{(i)}}$ is PSD, so all premises of the lemma hold. Next we show that for all i, the matrix $\mathbf{L}^{(i+1)}$ approximates $\mathbf{L}^{(i)}$ in the norm defined by $\mathbf{U}_{\mathbf{S}^{(i)}}$. By Lemma 7.3,

$$\mathbf{L}^{(i+1)} - \mathbf{L}^{(i)} = \begin{bmatrix} 0 & 0 \\ 0 & -\mathbf{C}_{q-i-1} \otimes c_{i+1} \mathbf{W}_{i+1} + \mathbf{C}_{q-i-1} \otimes c_{i+1} \mathbf{W}_i^2 \end{bmatrix}.$$

Now, given $c_{i+1} \mathbf{W}_{i+1} \overset{\circ}{\approx}_{\delta/40q^2} c_{i+1} \mathbf{W}_i^2$ by Lemma 5.13 and Corollary 5.10, from Lemmas 7.4 and 7.5 we obtain

$$\| \mathbf{U}^{+/2}_{\mathrm{Sc}(\mathbf{S}^{(i)}, H_i)} (\mathbf{L}^{(i+1)} - \mathbf{L}^{(i)}) \mathbf{U}^{+/2}_{\mathrm{Sc}(\mathbf{S}^{(i)}, H_i)} \|_2 \leq \delta / 40q^2$$

where H_i is the array of indices used for the $i+1$st Schur complement. Then, since $\mathbf{U}_{\mathrm{Sc}(\mathbf{S}^{(i)},H_i)} \preceq 2\mathbf{U}_{\mathbf{S}^{(i)}}$,

$$\|\mathbf{U}_{\mathbf{S}^{(i)}}^{+/2}(\mathbf{L}^{(i+1)} - \mathbf{L}^{(i)})\mathbf{U}_{\mathbf{S}^{(i)}}^{+/2}\|_2 \leq 2\delta/40q^2\,.$$

By construction, we have $\hat{\mathbf{S}}^{(a_i)} = \mathbf{S}^{(i)}$ and $\hat{\mathbf{M}}^{(a_i)} = \mathbf{L}^{(i)}$ for all $0 \leq i \leq q$. Therefore, we have

$$\|\mathbf{U}_{\hat{\mathbf{S}}^{(a_i)}}^{+/2}(\hat{\mathbf{M}}^{(a_i)} - \hat{\mathbf{M}}^{(a_{i+1})})\mathbf{U}_{\hat{\mathbf{S}}^{(a_i)}}^{+/2}\|_2 \leq 2\delta/40q^2\,.$$

Thus by Lemma 7.2 for $\mathbf{F} = \frac{2}{q}\sum_{i=0}^{q}\mathbf{U}_{\mathbf{S}^{(i)}}$ we obtain

$$\|\mathbf{F}^{+/2}(\mathbf{L} - \mathbf{L}^{(i)})\mathbf{F}^{+/2}\|_2 \leq \delta/40q \quad \forall 0 \leq i \leq q$$

and

$$\mathbf{L}^{(q)^T}\mathbf{F}^{+}\mathbf{L}^{(q)} \succeq \frac{1}{40q^2}\mathbf{F}\,. \qquad \square$$

8 Proofs of claims in the Introduction

We first show that weighted PRGs that consist entirely of positive coefficients are not substantially different from unweighted PRGs.

Proposition 8.1. *Given an explicit WPRG $(G,\rho) : \{0,1\}^s \to \{0,1\}^n \times \mathbb{R}$ such that $|\mathbb{E}_{x\leftarrow U_s}[\rho(x)] - 1| \leq \varepsilon$, there is an explicit PRG F with seed length $s' = s + O(\log(1/\varepsilon))$ such that for any function $B : \{0,1\}^n \to \{0,1\}$,*

$$|\mathbb{E}[B(F(U_{s'}))] - \mathbb{E}[\rho(U_s)\cdot B(G(U_s))]| \leq 2\varepsilon.$$

Note that if (G,ρ) is an ε-WPRG with all positive coefficients for a class of functions that contains the constant function $B(x) = 1$, we have $|\mathbb{E}_{x\leftarrow U_s}[\rho(x)\cdot 1] - 1| \leq \varepsilon$, so we can apply the above result and obtain that there exists an explicit 3ε-PRG for the class.

Finally, we give the proof of Proposition 8.1.

Proof. Let $\mu = \mathbb{E}_{x\leftarrow U_s}[\rho(x)]$ and define $\rho_T : \{0,1\}^s \to \mathbb{R}$ such that $\rho_T(x)$ is equal to $\rho(x)/\mu$ rounded up or down to a multiple of 2^{-d} while ensuring that $\mathbb{E}_{x\leftarrow U_s}[\rho_T(x)] = 1$. Choosing $d = \lceil\log(1/\varepsilon)\rceil$, we obtain that for arbitrary $B : \{0,1\}^n \to \{0,1\}$,

$$\begin{aligned}
\left|\mathop{\mathbb{E}}_{x\leftarrow U_s}[\rho(x)B(G(x))] - \mathop{\mathbb{E}}_{x\leftarrow U_s}[\rho_T(x)B(G(x))]\right| &= \left|\mathop{\mathbb{E}}_{x\leftarrow U_s}[\rho(x) - \rho_T(x)]\right| \\
&= \left|\mathop{\mathbb{E}}_{x\leftarrow U_s}[\rho(x) - \rho(x)/\mu + \rho(x)/\mu - \rho_T(x)]\right| \\
&\leq \left|\mathop{\mathbb{E}}_{x\leftarrow U_s}[\rho(x) - \rho(x)/\mu]\right| + \left|\mathop{\mathbb{E}}_{x\leftarrow U_s}[\rho(x)/\mu - \rho_T(x)]\right| \\
&\leq \varepsilon + \varepsilon.
\end{aligned}$$

So the weighted expectation of (G, ρ_T) is within 2ε of that of (G, ρ) on every boolean function. Then let $F : \{0,1\}^{s+d} \to \{0,1\}^n$ be an explicit PRG where for each seed $x \in \{0,1\}^s$, the output $G(x)$ appears with multiplicity $\rho_T(x) \cdot 2^d$ among the outputs $\{F(y)\}_{y \in \{0,1\}^{s+d}}$. Again fixing an arbitrary function B, we have

$$\begin{aligned}\mathop{\mathbb{E}}_{y \leftarrow U_{s+d}}[B(F(y))] &= \frac{1}{2^s \cdot 2^d} \sum_{y \in \{0,1\}^s \times \{0,1\}^d} B(F(y)) \\ &= \frac{1}{2^s} \sum_{x \in \{0,1\}^s} \frac{\rho_T(x) \cdot 2^d}{2^d} B(G(x)) = \mathop{\mathbb{E}}_{x \leftarrow U_s}[\rho_T(x) \cdot B(G(x))].\end{aligned}$$

So we obtain the desired result. The seed length is $s' = s + d = s + O(\log(1/\varepsilon))$. □

We next prove that an arbitrary explicit WPRG can be decomposed into a linear combination of two unweighted PRGs.

Proposition 8.2. *Given an explicit WPRG $(G, \rho) : \{0,1\}^s \to \{0,1\}^n \times \mathbb{R}$ and $\varepsilon > 0$, there are explicit generators $G_+ : \{0,1\}^{s'} \to \{0,1\}^n$ and $G_- : \{0,1\}^{s'} \to \{0,1\}^n$ with seed length $s' = O(s + \log(1/\varepsilon))$ and coefficients $\rho_+, \rho_- \in \mathbb{R}^{\geq 0}$ such that for every function $B : \{0,1\}^n \to \{0,1\}$, we have*

$$\left|\mathop{\mathbb{E}}_x[\rho(x) \cdot B(G(x))] - \left(\rho_+ \cdot \mathop{\mathbb{E}}_x[B(G_+(x))] - \rho_- \cdot \mathop{\mathbb{E}}_x[B(G_-(x))]\right)\right| \leq \varepsilon.$$

Proof. Let $\rho_+ = \mathbb{E}_{x \in U_s}[\rho(x) \cdot \mathbb{I}[\rho(x) \geq 0]]$ and $\rho_- = -\mathbb{E}_{x \in U_s}[\rho(x) \cdot \mathbb{I}[\rho(x) < 0]]$ be the average magnitude of the positive and negative coefficients, respectively (over the entire set of seeds). Then $R^+ = (G, \frac{\rho}{\rho_+}\mathbb{I}[\rho \geq 0])$ and $R^- = (G, -\frac{\rho}{\rho_-}\mathbb{I}[\rho < 0])$ are explicit WPRGs with all non-negative coefficients and expected weight exactly 1. We then apply Proposition 8.1 to R^+ with error $\varepsilon = \varepsilon/2\rho_+$ and obtain an unweighted WPRG $G_+ : \{0,1\}^{s'} \to \{0,1\}^n$ with seed length $s' = O(s + \log(1/\varepsilon))$ such that for every $B : \{0,1\}^n \to \{0,1\}$,

$$\left|\mathop{\mathbb{E}}_{x \leftarrow U_{s'}}[B(G_+(x))] - \mathop{\mathbb{E}}_{x \leftarrow U_s}\left[\frac{\rho(x)}{\rho_+}\mathbb{I}[\rho(x) \geq 0] B(G(x))\right]\right| \leq \frac{\varepsilon}{2\rho_+}.$$

Applying an identical transformation to R^- and applying the triangle inequality, we obtain for every $B : \{0,1\}^n \to \{0,1\}$,

$$\begin{aligned}&\left|\mathop{\mathbb{E}}_x[\rho(x) \cdot B(G(x))] - \left(\rho_+ \cdot \mathop{\mathbb{E}}_x[B(G_+(x))] - \rho_- \cdot \mathop{\mathbb{E}}_x[B(G_-(x))]\right)\right| \\ &\quad = \left|\left(\mathop{\mathbb{E}}_x[\rho(x)(\mathbb{I}[\rho(x) \geq 0] + \mathbb{I}[\rho(x) < 0]) \cdot B(G(x))]\right) - \left(\rho_+ \cdot \mathop{\mathbb{E}}_x[B(G_+(x))] - \rho_- \cdot \mathop{\mathbb{E}}_x[B(G_-(x))]\right)\right| \\ &\quad \leq \left|\mathop{\mathbb{E}}_x[\rho(x)\mathbb{I}[\rho(x) \geq 0] \cdot B(G(x))] - \rho_+ \cdot \mathop{\mathbb{E}}_x[B(G_+(x))]\right| \\ &\qquad + \left|\rho_- \cdot \mathop{\mathbb{E}}_x[B(G_-(x))] - \mathop{\mathbb{E}}_x[\rho(x)\mathbb{I}[\rho(x) < 0] \cdot B(G(x))]\right| \\ &\quad \leq \rho_+\left(\frac{\varepsilon}{2\rho_+}\right) + \rho_-\left(\frac{\varepsilon}{2\rho_-}\right).\end{aligned}$$

Furthermore, note that if (G, ρ) ε-fools the function that accepts on all inputs, it must be the case that $\rho_+ - \rho_- \in [1-\varepsilon, 1+\varepsilon]$ and so we can take $\rho_- = 1 - \rho_+$ at the cost of an additive $O(\varepsilon)$ loss in approximation. □

We next prove that a WPRG for a class that includes indicator test functions must have bounded coefficients. We remark that permutation branching programs of unbounded width have this property, whereas permutation branching programs of width less than n do not.

Proposition 8.3. *Given an explicit ε-WPRG $(G, \rho) : \{0,1\}^s \to \{0,1\}^n \times \mathbb{R}$ that fools a class $\mathcal{F}$ of functions that includes the indicator functions $\mathbb{I}[x = x_0]$ for every x_0, we have that ρ is $(\varepsilon + 2^{-n})2^s$-bounded.*

Proof. First, we assume that $G(x) \neq G(x')$ for every $x \neq x'$, since otherwise we could combine $\rho(x), \rho(x')$ and decrease the seed length. Fix an arbitrary $x_0 \in \{0,1\}^s$ and let $y = G(x_0)$, and consider the function $f(z) = \mathbb{I}[z = y]$. As $f \in \mathcal{F}$, we have

$$\begin{aligned} \varepsilon &\geq \left| \mathbb{E}[f(U_n)] - \mathop{\mathbb{E}}_{x \leftarrow U_s}[\rho(x) f(G(x))] \right| \\ &= \left| 2^{-n} - \frac{\rho(x_0)}{2^s} \cdot f(G(x_0)) \right| \end{aligned}$$

and thus $(\varepsilon + 2^{-n})2^s \geq |\rho(x)|$. □

Finally, we give a contrived example of a class for which WPRGs obtain exponentially shorter seed length than any PRG. We do this by requiring that the class satisfy a functional equation that WPRGs can exploit but PRGs cannot.

Proposition 8.4. *Let $\mathcal{B}$ be the set of all functions $B : \{0,1\}^n \to \{0,1\}$ such that*

$$\mathbb{E}[B(U_n)] = \mathbb{E}[B(U_S)] - \mathbb{E}[B(U_T)], \tag{8.1}$$

where $S = \{0^{n-\log n} z : z \in \{0,1\}^{\log n}\}$ and $T = \{1^{n-\log n} z : z \in \{0,1\}^{\log n}\}$. Then there exists an explicit 0-WPRG for $\mathcal{B}$ with seed length $O(\log n)$. Furthermore, any $(1-1/n)$-PRG for $\mathcal{B}$ has seed length $\Omega(n)$.

Proof. Let $(G, \rho) : \{0,1\}^{1+\log n} \to \{0,1\}^n \times \mathbb{R}$ be the WPRG where $\rho(x) = 2$ if $x_1 = 1$ and $\rho(x) = -2$ if $x_1 = 0$ and $G(x) = 0^{n-\log n} x_{2..\log n+1}$ if $x_1 = 1$ and $G(x) = 1^{n-\log n} x_{2..\log n+1}$ if $x_1 = 0$. Then for every $B \in \mathcal{B}$,

$$\begin{aligned} \left| \mathop{\mathbb{E}}_{y \leftarrow U_{1+\log n}}[\rho(y) \cdot B(G(y))] - \mathop{\mathbb{E}}_{y \leftarrow U_n}[B(y)] \right| &= \left| \left(\mathop{\mathbb{E}}_{y \leftarrow U_S}[B(y)] - \mathop{\mathbb{E}}_{y \leftarrow U_T}[B(y)] \right) - \mathop{\mathbb{E}}_{y \leftarrow U_n}[B(y)] \right| \\ &= 0\,. \end{aligned}$$

Now consider an arbitrary PRG $F : \{0,1\}^s \to \{0,1\}^n$ with seed length $s \leq n - 2\log(n)$. Let $\mathrm{Im}(F)$ be the image of F. Now choose some $x^* \in T$ where $\Pr[F(U_s) = x^*] \leq 1/|T|$, which is always possible since $\min_{x \in T} \Pr[F(U_s) = x] \cdot |T| \le \sum_{x \in T} \Pr[F(U_s) = x] \le 1$. We now define the function B such that $B(x) = 1$ for all $x \in M = (S \cup T \cup \mathrm{Im}(F)) \setminus \{x^*\}$. Note that $|\{0,1\}^n \setminus M| \ge (1 - 3/n)2^n$. We have that $\mathbb{E}[B(U_S)] - \mathbb{E}[B(U_T)] = 1 - (1 - 1/n) = 1/n$, so set $B(x) = 1$ at a sufficient number of points in $\{0,1\}^n \setminus M$ so that B satisfies Equation (8.1), and otherwise set $B(x) = 0$. By construction, B satisfies Equation (8.1) with $\mathbb{E}[B(U_n)] = 1/n$. However, $\mathbb{E}[B(F(U_s))] = 1$, so F is not a $(1 - 1/n)$-PRG for the class. □

We now prove facts about samplers for functions with bounded variance. Such functions form a natural class where general nonadaptive samplers obtain smaller sampler complexity than averaging samplers.

Definition 8.5. An (ε, δ)-nonadaptive sampler G for a class $\mathcal{F}$ of functions is a randomized function that makes nonadaptive queries to $f \in \mathcal{F}$ and returns as estimate ρ such that, over the randomness of the algorithm,

$$\Pr[|\rho - \mathbb{E}[f]| \le \varepsilon] \ge 1 - \delta.$$

We say G is an **averaging sampler** if G generates (possibly correlated) points $x_1, \ldots, x_t$ in the domain of f and returns $\rho = \frac{1}{t} \sum_{i=1}^{t} f(x_i)$.

We then define the model to study.

Definition 8.6. Let $\mathcal{F} = \{f : \{0,1\}^m \to \mathbb{R} : \mathrm{Var}(f) \le 1\}$ be the set of unbounded functions with variance at most 1.

We first prove that samplers exist with the claimed parameters.

Proposition 8.7. *There is an averaging sampler for $\mathcal{F}$ with sample complexity* $\min\{2^m, O(1/\varepsilon^2\delta)\}$, *and a nonadaptive sampler with sample complexity* $\min\{2^m, O(\log(1/\delta)/\varepsilon^2)\}$.

Proof. Fixing $\varepsilon, \delta > 0$, first note that if either minimum is 2^m the relevant statement is immediate by querying on all points of $\{0,1\}^m$ and returning the average, which is exactly the expectation of the function.

For the averaging sampler, let $t = 1/\varepsilon^2\delta$. Now fix an arbitrary $f \in \mathcal{F}$. The sampler generates t independent points $X_1, \ldots, X_t$ from U_m. Let $Y_1, \ldots, Y_t$ be the random variables where $Y_i = f(X_i)$, and define $Y = \frac{1}{t} \sum_{i=1}^{t} Y_i$. Then $\mathbb{E}[Y] = \mathbb{E}[f(U_m)]$ via linearity of expectation and $\mathrm{Var}(Y) = \frac{1}{t^2} \sum_{i=1}^{t} \mathrm{Var}(Y_i) = 1/t$ by independence. Applying Chebyshev's Inequality to Y, we obtain, for any $k > 0$,

$$\Pr\left[|Y - \mathbb{E}[f(U_m)]| \ge \frac{k}{\sqrt{t}}\right] \le \frac{1}{k^2}$$

and choosing $k = 1/\sqrt{\delta}$ we obtain

$$\Pr\left[|Y - \mathbb{E}[f(U_m)]| \ge \frac{1}{\sqrt{\delta}} \varepsilon \sqrt{\delta}\right] \le \delta\,,$$

exactly the required condition for the averaging sampler.

For the nonadaptive sampler, choose $t = 1/10\varepsilon^2$ and $T = 10\log(1/\delta)$. The sampler generates $X_{i,j} \leftarrow U_m$ for $i \in [t]$ and $j \in [T]$, and we define the random variables $Y_{i,j} = f(X_{i,j})$. Let $Y_j = \frac{1}{t}\sum_{i=1}^{t} Y_{i,j}$ for all j. Then by Chebyshev, for all j,

$$\Pr[|Y_j - \mathbb{E}[f(U_m)]| \geq \varepsilon] \leq 1/10.$$

Note that the Y_j are independent and have mean $\mathbb{E}[f(U_m)]$ for all j. Then define the (independent) indicator variables $B_j = \mathbb{I}[|Y_j - \mathbb{E}[f(U_m)]| \geq \varepsilon]$ and note that $\mathbb{E}[B_j] = \Pr[|Y_j - \mathbb{E}[f(U_m)]| \geq \varepsilon] \leq 1/10$ for all j. If $\sum_{j=1}^{T} B_j < T/3$, we have that the median of the Y_j is within ε of $\mathbb{E}[f(U_m)]$, so the sampler succeeds. We bound the probability of this failing to occur by Bernstein–Hoeffding where for $\rho > 0$ we have

$$\Pr\left[\sum_{j=1}^{T} B_j \geq T/3\right] \leq \Pr\left[\sum_{j=1}^{T} B_j - T/10 \geq (T/10)\rho\right] \leq \exp(-\rho^2 T/10(2+\rho)) \leq \delta\,,$$

where the final step follows from choosing $\rho = 2$ and recalling $T = 10\log(1/\delta)$. □

Finally, we prove a lower bound on sampler length for averaging samplers.

Proposition 8.8. *Let A be an (ε, δ) averaging sampler for $\mathcal{F}$. Then A makes $t = \min\{\widetilde{\Omega}(1/\varepsilon^2\delta), 2^{\Omega(m)}\}$ queries.*

Proof. We first define our set of test functions. For $1 \geq \rho \geq 4/2^m$ to be chosen later, let $\mathcal{F}_\rho \subset \mathcal{F}$ be the class of functions obtained by selecting sets $S^- \subseteq \{0,1\}^m$ and $S^+ \subseteq \{0,1\}^m \setminus S^-$ both of size $\lfloor(\rho/2)\cdot 2^m\rfloor$ uniformly at random without replacement, and setting

$$f_{S^-,S^+}(x) = \begin{cases} 1/\sqrt{\rho} & x \in S^+ \\ -1/\sqrt{\rho} & x \in S^- \\ 0 & \text{otherwise.} \end{cases}$$

For all $f \in \mathcal{F}_\rho$ we obtain $\mathbb{E}[f(U_m)] = 0$ and $\mathrm{Var}(f) \leq (1/\sqrt{\rho})^2\rho = 1$.

If $t \geq 2^{m/3-1}$ the statement is immediately true, so we assume this is not the case. Furthermore, if $\delta \leq 2^{-m/3}$ we set $\delta = 2^{-m/3}$, which does not affect the asymptotic lower bound, and likewise if $t \geq 2^{1/2\delta}$ we set $\delta = 1/2\log t$ (as decreasing δ cannot affect the bound).

For every random seed σ for the sampler, there is $k(\sigma) \in \{1,\ldots,\log t\}$ such that at least $t/\log t$ of the (not necessarily distinct) points queried by A are queried with multiplicity $\{2^{k(\sigma)-1},\ldots,2^{k(\sigma)}\}$ by the pigeonhole principle. Furthermore, there is $k \in \{1,\ldots,\log t\}$ such that for at least a $1/\log t$ fraction of the seeds, $k(\sigma) = k$, again via pigeonhole.

Let $C(\sigma)$ be the event that there are at least $t/2^k \log t$ *distinct* points queried by A with multiplicity at least 2^{k-1}. We have $\Pr_\sigma[C(\sigma)] \geq 1/\log t$ as with probability $1/\log t$ over the random seed at least $t/\log t$ of the total points are queried with multiplicities in the range $\{2^{k-1}, \dots, 2^k\}$, so there are at least $t/2^k \log t$ distinct points with the desired property.

We now choose $\rho = \delta \log^2(t)2^k/t$ (which is valid as $\rho \geq \delta/t \geq 2^{-2m/3}$ and $\rho \leq \delta \log^2 t \leq 1$) and consider the behavior of the sampler on f drawn uniformly at random from $\mathcal{F}_\rho$. Fix σ for which $C(\sigma)$ occurs and let $x_1, \dots, x_t$ be the multiset of points queried by A. Let $B(\sigma, f)$ be the event that f takes a nonzero value on at least one point queried with multiplicity at least 2^{k-1}. We have

$$\begin{aligned}
\Pr_{\sigma,f}[B(\sigma, f)] &\geq \Pr_{\sigma}[C(\sigma)] \cdot \left[1 - (1-\rho)^{t/2^k \log^2 t}\right] \\
&\geq \Pr_{\sigma}[C(\sigma)] \cdot \left[(\rho)(t/2^k \log t) - \rho^2 (t/2^k \log t)^2\right] \\
&\geq (1/\log t)(\delta \log(t)/2) \\
&= \delta/2
\end{aligned}$$

where the first line follows because this event becomes strictly less likely if the nonzero points of f are sampled independently with probability ρ, the second follows from two rounds of inclusion-exclusion, and the third from the assumption that $\rho(t/2^k \log t) \leq \delta \log t \leq 1/2$.

Conditioned on $B(\sigma, f)$ occurring, WLOG assume x_1 is queried with multiplicity at least 2^{k-1} and $f(x_1) \neq 0$. Letting $Y = \sum_{i=1:x_i \neq x_1}^{t} f(x_i)$ be the sum of the queried points excluding x_1, we claim

$$\Pr_f[f(x_1) \geq 0, Y \geq 0 | B(\sigma, f)] \geq 1/8.$$

As $B(\sigma, f)$ is independent of the sign of f on all nonzero points, $\Pr_f[f(x_1) \geq 0 | B(\sigma, f)] = 1/2$. Since $t \leq 2^{m/3-1}$ and there are at least $2^m(\delta/t) \geq 2^{m/3}$ nonzero points for all $f \in \mathcal{F}_\rho$, for every f there is some point x' not queried by A where $f(x') \neq 0$, and

$$\Pr_f[f(x') < 0 | B(\sigma, f), f(x_1) > 0] \geq \frac{(\rho/2)2^m + 1}{\rho 2^m} \geq 1/2.$$

Then, since we condition on one point of each sign, $\mathbb{E}_f[Y | f(x_1) > 0, f(x') < 0] = 0$ and the distribution is symmetric, so $\Pr_f[f(x_1) \geq 0, Y \geq 0 | B(f, \sigma)] \geq 1/4$. Thus with probability at least $\delta/16$,

$$\left| \frac{1}{t} \sum_{i=1}^{t} f(x_i) - \mathbb{E}[f] \right| \geq \frac{2^{k-1}}{t} f(x_1) = \frac{2^{k-1}}{t} \sqrt{\frac{t}{2^k \delta \log^2 t}}.$$

By taking $\delta \leftarrow 20\delta$ we can obtain that this event occurs with probability strictly greater than δ, and so by the fact that A is an (ε, δ) sampler we obtain the constraint

$$2\varepsilon\sqrt{20\delta} \geq \frac{1}{\sqrt{(t/2^k)\log^2 t}} \geq \frac{1}{\sqrt{t \log^2 t}}$$

and thus derive $t \log^2 t \geq 1/80\varepsilon^2\delta$, which is implied by $t = \Omega(1/\varepsilon^2 \delta \log^2(1/\varepsilon\delta))$ (with a sufficiently small constant). ◻

9 Acknowledgements

We thank Jack Murtagh and Sumegha Garg for insightful discussions, and Oded Goldreich and the CCC'21 and *Theory of Computing* reviewers for feedback that improved our presentation.

References

[1] ROHIT AGRAWAL: Samplers and extractors for unbounded functions. In *Proc. 23rd Internat. Conf. on Randomization and Computation (RANDOM'19)*, pp. 59:1–21. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. [doi:10.4230/LIPIcs.APPROX-RANDOM.2019.59] 16

[2] AMIRMAHDI AHMADINEJAD, JONATHAN A. KELNER, JACK MURTAGH, JOHN PEEBLES, AARON SIDFORD, AND SALIL P. VADHAN: High-precision estimation of random walks in small space. In *Proc. 61st FOCS*, pp. 1295–1306. IEEE Comp. Soc., 2020. [doi:10.1109/FOCS46700.2020.00123] 7, 9, 12, 16, 22, 23, 28, 31, 33, 34, 38, 39, 49, 50, 51, 52

[3] ALEXANDER E. ANDREEV, ANDREA E. F. CLEMENTI, AND JOSÉ D. P. ROLIM: A new general derandomization method. *J. ACM*, 45(1):179–213, 1998. [doi:10.1145/273865.273933] 16

[4] ALEXANDER E. ANDREEV, ANDREA E. F. CLEMENTI, JOSÉ D. P. ROLIM, AND LUCA TREVISAN: Weak random sources, hitting sets, and BPP simulations. *SIAM J. Comput.*, 28(6):2103–2116, 1999. [doi:10.1137/s0097539797325636] 16

[5] ROY ARMONI: On the derandomization of space-bounded computations. In *Proc. 2nd Internat. Workshop on Randomization and Computation (RANDOM'98)*, pp. 47–59. Springer, 1998. [doi:10.1007/3-540-49543-6_5] 4, 7

[6] JAROSLAW BLASIOK: Optimal streaming and tracking distinct elements with high probability. *ACM Trans. Algorithms*, 16(1):3:1–28, 2019. Preliminary version in SODA'18. [doi:10.1145/3309193] 16

[7] MANUEL BLUM AND SILVIO MICALI: How to generate cryptographically strong sequences of pseudorandom bits. *SIAM J. Comput.*, 13(4):850–864, 1984. [doi:10.1137/0213053] 2

[8] ANDREJ BOGDANOV, ZEEV DVIR, ELAD VERBIN, AND AMIR YEHUDAYOFF: Pseudorandomness for width-2 branching programs. *Theory of Computing*, 9(7):283–293, 2013. [doi:10.4086/toc.2013.v009a007, ECCC:TR09-070] 4

[9] ANDREJ BOGDANOV, WILLIAM M. HOZA, GAUTAM PRAKRIYA, AND EDWARD PYNE: Hitting sets for regular branching programs. In *Proc. 37th Comput. Complexity Conf. (CCC'22)*, pp. 3:1–22. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2022. [doi:10.4230/LIPIcs.CCC.2022.3] 7

[10] MARK BRAVERMAN, GIL COHEN, AND SUMEGHA GARG: Hitting sets with near-optimal error for read-once branching programs. In *Proc. 50th STOC*, pp. 353–362. ACM Press, 2018. [doi:10.1145/3188745.3188780] 2, 3, 4, 6, 8, 11

[11] MARK BRAVERMAN, ANUP RAO, RAN RAZ, AND AMIR YEHUDAYOFF: Pseudorandom generators for regular branching programs. *SIAM J. Comput.*, 43(3):973–986, 2014. Preliminary version in FOCS'10. [doi:10.1137/120875673] 5, 6, 7

[12] HARRY BUHRMAN AND LANCE FORTNOW: One-sided versus two-sided error in probabilistic computation. In *Proc. 16th Symp. Theoret. Aspects of Comp. Sci. (STACS'99)*, pp. 100–109. Springer, 1999. [doi:10.1007/3-540-49116-3_9] 16

[13] ESHAN CHATTOPADHYAY AND JYUN-JIE LIAO: Optimal error pseudodistributions for read-once branching programs. In *Proc. 35th Comput. Complexity Conf. (CCC'20, virtual conference)*, pp. 25:1–27. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020. [doi:10.4230/LIPIcs.CCC.2020.25] 4, 6, 8, 11, 26

[14] ESHAN CHATTOPADHYAY AND JYUN-JIE LIAO: Recursive Error Reduction for Regular Branching Programs. In *Proc. 15th Innovations in Theoret. Comp. Sci. Conf. (ITCS'24)*, pp. 29:1–20. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2024. [doi:10.4230/LIPIcs.ITCS.2024.29] 7

[15] LIJIE CHEN, WILLIAM M. HOZA, XIN LYU, AVISHAY TAL, AND HONGXUN WU: Weighted pseudorandom generators via inverse analysis of random walks and shortcutting. In *Proc. 64th FOCS*, pp. 1224–1239. IEEE Comp. Soc., 2023. [doi:10.1109/FOCS57990.2023.00072] 7

[16] KUAN CHENG AND WILLIAM M. HOZA: Hitting sets give two-sided derandomization of small space. *Theory of Computing*, 18(21):1–32, 2022. Preliminary version in CCC'20. [doi:10.4086/toc.2022.v018a021] 16

[17] GIL COHEN, DEAN DORON, OREN RENARD, ORI SBERLO, AND AMNON TA-SHMA: Error reduction for weighted PRGs against Read Once Branching Programs. In *Proc. 36th Comput. Complexity Conf. (CCC'21)*, pp. 22:1–17. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2021. [doi:10.4230/LIPIcs.CCC.2021.22] 3, 6, 8

[18] MICHAEL B. COHEN, JONATHAN KELNER, RASMUS KYNG, JOHN PEEBLES, RICHARD PENG, ANUP B. RAO, AND AARON SIDFORD: Solving directed Laplacian systems in nearly-linear time through sparse LU factorizations. In *Proc. 59th FOCS*, pp. 898–909. IEEE Comp. Soc., 2018. [doi:10.1109/FOCS.2018.00089] 51

[19] 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 *Proc. 49th STOC*, pp. 410–419. ACM Press, 2017. [doi:10.1145/3055399.3055463] 16, 38

[20] ANINDYA DE: Pseudorandomness for permutation and regular branching programs. In *Proc. 26th IEEE Conf. on Comput. Complexity (CCC'11)*, pp. 221–231. IEEE Comp. Soc., 2011. [doi:10.1109/CCC.2011.23] 5, 6

[21] ODED GOLDREICH: *A primer on pseudorandom generators*. Volume 55 of *University Lecture Series*. Amer. Math. Soc., 2010. Link at AMS. Accessible at The Weizmann Institute. 2

[22] ODED GOLDREICH: A sample of samplers: A computational perspective on sampling. In ODED GOLDREICH, editor, *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, volume 6650 of *LNCS*, pp. 302–332. Springer, 2011. [doi:10.1007/978-3-642-22670-0_24, ECCC:TR97-020] 15

[23] ODED GOLDREICH, SALIL VADHAN, AND AVI WIGDERSON: Simplified derandomization of BPP using a hitting set generator. In ODED GOLDREICH, editor, *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, volume 6650 of *LNCS*, pp. 59–67. Springer, 2011. [doi:10.1007/978-3-642-22670-0_8] 16

[24] PARIKSHIT GOPALAN, RAGHU MEKA, OMER REINGOLD, LUCA TREVISAN, AND SALIL VADHAN: Better pseudorandom generators from milder pseudorandom restrictions. In *Proc. 53rd FOCS*, pp. 120–129. IEEE Comp. Soc., 2012. [doi:10.1109/focs.2012.77] 4

[25] WILLIAM M. HOZA: Better pseudodistributions and derandomization for space-bounded computation. In *Proc. 25th Internat. Conf. on Randomization and Computation (RANDOM'21)*, pp. 28:1–23. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2021. [doi:10.4230/LIPIcs.APPROX/RANDOM.2021.28] 6

[26] WILLIAM M. HOZA, EDWARD PYNE, AND SALIL P. VADHAN: Pseudorandom generators for unbounded-width permutation branching programs. In *Proc. 12th Innovations in Theoret. Comp. Sci. Conf. (ITCS'21)*, pp. 7:1–20. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2021. [doi:10.4230/LIPIcs.ITCS.2021.7] 5, 6, 14, 15, 16, 28, 29, 30, 31, 32, 45, 46

[27] RUSSELL IMPAGLIAZZO, NOAM NISAN, AND AVI WIGDERSON: Pseudorandomness for network algorithms. In *Proc. 26th STOC*, pp. 356–364. ACM Press, 1994. [doi:10.1145/195058.195190] 6, 11, 26

[28] PIOTR INDYK: Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, 2006. [doi:10.1145/1147954.1147955] 4

[29] DANIEL M. KANE, JELANI NELSON, AND DAVID P. WOODRUFF: Revisiting norm estimation in data streams, 2008. [arXiv:0811.3648] 4

[30] EYAL KAPLAN, MONI NAOR, AND OMER REINGOLD: Derandomized constructions of k-wise (almost) independent permutations. *Algorithmica*, 55(1):113–133, 2009. Preliminary version in RANDOM'05. [doi:10.1007/s00453-008-9267-y] 4

[31] MICHAL KOUCKÝ, PRAJAKTA NIMBHORKAR, AND PAVEL PUDLÁK: Pseudorandom generators for group products: Extended abstract. In *Proc. 43rd STOC*, pp. 263–272. ACM Press, 2011. [doi:10.1145/1993636.1993672] 5, 6

[32] RAGHU MEKA, OMER REINGOLD, AND AVISHAY TAL: Pseudorandom generators for width-3 branching programs. In *Proc. 51st STOC*, pp. 626–637. ACM Press, 2019. [doi:10.1145/3313276.3316319] 4

[33] NOAM NISAN: Pseudorandom bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991. [doi:10.1007/BF01375474] 2

[34] NOAM NISAN: Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992. [doi:10.1007/bf01305237] 4, 6, 7, 20

[35] NOAM NISAN AND AVI WIGDERSON: Hardness vs. randomness. *J. Comput. System Sci.*, 49(2):149–167, 1994. Preliminary version in FOCS'88. [doi:10.1016/S0022-0000(05)80043-1] 2

[36] NOAM NISAN AND DAVID ZUCKERMAN: Randomness is linear in space. *J. Comput. System Sci.*, 52(1):43–52, 1996. Preliminary version in STOC'93. See updated version at author's website. [doi:10.1006/jcss.1996.0004] 4

[37] EDWARD PYNE AND SALIL VADHAN: Pseudodistributions that beat all pseudorandom generators. *Electron. Colloq. Comput. Complexity*, TR21-019, 2021. [ECCC] 3

[38] EDWARD PYNE AND SALIL VADHAN: Pseudodistributions that beat all pseudorandom generators (Extended abstract). In *Proc. 36th Comput. Complexity Conf. (CCC'21)*, pp. 33:1–15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2021. [doi:10.4230/LIPIcs.CCC.2021.33] 1

[39] OMER REINGOLD, THOMAS STEINKE, AND SALIL VADHAN: Pseudorandomness for regular branching programs via Fourier analysis. In *Proc. 17th Internat. Workshop on Randomization and Computation (RANDOM'13)*, pp. 655–670. Springer, 2013. [doi:10.1007/978-3-642-40328-6_45] 17

[40] OMER REINGOLD, LUCA TREVISAN, AND SALIL VADHAN: Pseudorandom walks on regular digraphs and the RL vs. L problem. In *Proc. 38th STOC*, pp. 457–466. ACM Press, 2006. [doi:10.1145/1132516.1132583, ECCC:TR05-022] 5

[41] EYAL ROZENMAN AND SALIL VADHAN: Derandomized squaring of graphs. In *Proc. 9th Internat. Workshop on Randomization and Computation (RANDOM'05)*, pp. 436–447. Springer, 2005. [doi:10.1007/11538462_37] 5, 13, 14, 32

[42] MICHAEL SAKS AND SHIYU ZHOU: $\mathrm{BP_HSPACE}(S) \subseteq \mathrm{DSPACE}(S^{3/2})$. *J. Comput. System Sci.*, 58(2):376–403, 1999. Preliminary version in FOCS'95. [doi:10.1006/jcss.1998.1616] 7, 9

[43] JIRÍ SÍMA AND STANISLAV ZÁK: Almost k-wise independent sets establish hitting sets for width-3 1-branching programs. In *Proc. Comp. Sci. Symp. in Russia (CSR'11)*, pp. 120–133. Springer, 2011. [doi:10.1007/978-3-642-20712-9_10] 4

[44] THOMAS STEINKE: Pseudorandomness for permutation branching programs without the group theory. *Electron. Colloq. Comput. Complexity*, TR12-083, 2012. [ECCC] 5, 6

[45] SALIL P. VADHAN: Pseudorandomness. *Found. Trends Theor. Comp. Sci.*, 7(1–3):1–336, 2012. [doi:10.1561/0400000010] 2, 16

[46] ANDREW C. YAO: Theory and applications of trapdoor functions (extended abstract). In *Proc. 23rd FOCS*, pp. 80–91. IEEE Comp. Soc., 1982. [doi:10.1109/sfcs.1982.45] 2

[47] DAVID ZUCKERMAN: Randomness-optimal oblivious sampling. *Random Struct. Algor.*, 11(4):345–367, 1997. [doi:10.1002/(sici)1098-2418(199712)11:4<345::aid-rsa4>3.0.co;2-z] 16

AUTHORS

Edward Pyne
Ph. D. Student
Computer Science and AI Lab
MIT
Cambridge, MA, USA
epyne@mit.edu
https://sites.google.com/view/tedpyne/

Salil Vadhan
Vicky Joseph Professor of Computer Science and Applied Mathematics
Department of Computer Science
Harvard University
Cambridge, MA, USA
salil_vadhan@harvard.edu
http://salil.seas.harvard.edu/

ABOUT THE AUTHORS

Edward Pyne is a Ph. D. student at MIT advised by Ronitt Rubinfeld. He previously was an undergraduate at Harvard University, where he was advised by Salil Vadhan. His research interests include pseudorandomness, catalytic computing, and sublinear algorithms. He has visited 38 countries and 47 US states.

Salil Vadhan is the Vicky Joseph Professor of Computer Science and Applied Mathematics at the Harvard John A. Paulson School of Engineering & Applied Sciences. He received his Ph. D. under the supervision of Shafi Goldwasser at MIT in 1999; the title of his dissertation was "A Study of Statistical Zero-Knowledge Proofs." Other research interests include the theory of pseudorandomness and the theory and practice of data privacy. He enjoys spending leisure time with his wife and two daughters, as well as learning to surf in the cold waters of New England.