Electronic Colloquium on Computational Complexity, Report No. 19 (2021)

Pseudodistributions That Beat All Pseudorandom Generators

Edward Pyne*
Harvard University
epyne@college.harvard.edu

Salil Vadhan†
Harvard University
salil_vadhan@harvard.edu

February 17, 2021

Abstract

A recent paper of Braverman, Cohen, and Garg (STOC 2018) introduced the concept of a *pseudorandom pseudodistribution generator (PRPG)*, which amounts to a pseudorandom generator (PRG) whose outputs are accompanied with real coefficients that scale the acceptance probabilities of any potential distinguisher. They gave an explicit construction of PRPGs for ordered branching programs whose seed length has a better dependence on the error parameter ε than the classic PRG construction of Nisan (STOC 1990 and Combinatorica 1992).

In this work, we give an explicit construction of PRPGs that achieve parameters that are *impossible* to achieve by a PRG. In particular, we construct a PRPG for *ordered permutation branching programs of unbounded width* with a single accept state that has seed length $\tilde{O}(\log^{3/2} n)$ for error parameter $\varepsilon = 1/\operatorname{poly}(n)$, where n is the input length. In contrast, recent work of Hoza et al. (ITCS 2021) shows that any PRG for this model requires seed length $\Omega(\log^2 n)$ to achieve error $\varepsilon = 1/\operatorname{poly}(n)$.

As a corollary, we obtain explicit PRPGs with seed length $\tilde{O}(\log^{3/2} n)$ and error $\varepsilon = 1/\operatorname{poly}(n)$ for ordered permutation branching programs of width $w = \operatorname{poly}(n)$ 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 2010 and SICOMP 2014).

The starting point for our results are the recent space-efficient algorithms for estimating random-walk probabilities in directed graphs by Ahmadenijad, Kelner, Murtagh, Peebles, Sidford, and Vadhan (FOCS 2020), which are based on spectral graph theory and space-efficient Laplacian solvers. We interpret these algorithms as giving PRPGs 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 (personal communication, January 2021).

Keywords: pseudorandomness, space-bounded computation, spectral graph theory

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

ISSN 1433-8092

1 Introduction

The notion of a **pseudorandom generator (PRG)** [BM84, Yao82, NW94] is ubiquitous in theoretical computer science, with vast applicability in cryptography and derandomization. A recent work of Braverman, Cohen, and Garg [BCG18] 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 functions $B\colon \{0,1\}^n \to \{0,1\}$. An **$\varepsilon$-pseudorandom pseudodistribution generator (PRPG)**, or **pseudorandom pseudogenerator** for $\mathcal{B}$ is a function $(G,\rho)\colon \{0,1\}^s \to \{0,1\}^n \times \mathbb{R}$ such that for every $B \in \mathcal{B}$,

$$\left| \mathop{\mathbb{E}}_{x \leftarrow U_{\{0,1\}^n}} [B(x)] - \mathop{\mathbb{E}}_{x \leftarrow U_{\{0,1\}^s}} [\rho(x) \cdot B(G(x))] \right| \leq \varepsilon.$$

The value s is the **seed length** of the PRPG, and n is the **output length** of the PRPG. If $|\rho(x)| \leq r$ for all x we say the PRPG is **r-bounded**. We say that the PRPG is **explicit** if given x, $G(x)$ and $\rho(x)$ are computable in space $O(s)$, and $|\rho(x)|$ is bounded by $2^{O(s)}$.

(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.) With this definition, a PRG is a special case of a PRPG with $\rho(x) = 1$. As noted in [BCG18], PRPGs 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 PRPG (G,ρ) and calculating the average of the values $\rho(x) \cdot B(G(x))$. Furthermore, [BCG18] observe that if (G,ρ) is an ε-PRPG for a model then G is an **ε-hitting set generator (HSG)**. 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 $B(G(x)) = 1$.

1.1 Ordered Branching Programs

The work of Braverman, Cohen, and Garg [BCG18], as well as our paper, focuses on PRPGs 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 $B_1, \ldots, B_n$, the start state v_0 and the set V_{acc} of accept states.

An ordered branching program of length n and width w can compute the output of an algorithm that uses $\log w$ bits of memory and n random bits, by taking the state at each layer as the contents of memory at that time. 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 [Nis92] gives an explicit PRG with seed length $s = O(\log n \cdot \log(nw/\varepsilon))$, and this bound has not been improved except when $w \leq 3$ [BDVY09, SZ11, GMR+12, MRT19]. Braverman, Cohen, and Garg [BCG18] gave an explicit construction of a PRPG that achieves improved dependence on the error parameter ε, with seed length

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

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 $\tilde{O}(\log^2 n)$. Chatthopadhyay and Liao [CL20] gave a simpler construction of PRPGs with a slightly shorter seed length than [BCG18], with an additive dependence on $O(\log(1/\varepsilon))$ rather than $\tilde{O}(\log(1/\varepsilon))$.

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, attention has turned to more restricted classes of ordered branching programs. In this work, 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.

Previous works on various types of PRGs for permutation branching programs [RV05, RTV06, BRRY10, KNP11, De11, Ste12, HPV21] have 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. Specifically, Braverman, Rao, Raz, and Yehudayoff gave a PRG for the more general model of *regular* branching programs with seed length

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

For the specific case of permutation branching programs, Koucký, Nimbhorkar, and Pudlák [KNP11], De [De11], and Steinke [Ste12] showed how to remove the $\log\log n$ term at the price of a worse dependence on w, achieving seed length

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

Most recently, Hoza, Pyne, and Vadhan [HPV21] showed that the dependence on the width w could be entirely eliminated if we restrict to permutation branching programs with a *single accept state*, achieving 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. 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 PRPG for permutation branching programs of unbounded width and a single accept state that beats the aforementioned lower bounds for PRGs:

Theorem 1.4. *For all $n \in N$ and $\varepsilon \in (0, 1/2)$, there is an explicit ε-PRPG 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 $\varepsilon = 1/\operatorname{poly}(n)$, we achieve seed length $\tilde{O}(\log^{3/2} n)$, while a PRG requires seed length $\Omega(\log^2 n)$ [HPV21].

As noted in [HPV21], an ε-PRPG for branching programs with a single accept state is also an $(a \cdot \varepsilon)$-PRPG for branching programs with at most a accept states. For bounded-width permutation branching programs, we can take $a = w$ and obtain:

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

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

In particular for $w = \operatorname{poly}(n)$ and $\varepsilon = 1/\operatorname{poly}(n)$, we achieve seed length $\tilde{O}(\log^{3/2} n)$. 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 $w = n^{\tilde{\Omega}(\log n)}$ and error as small as $\varepsilon = 1/n^{-\tilde{\Omega}(\log(n))}$. We summarize these results in a table.

Citation	Type	Model	Seed Length
Non-explicit (folklore)	PRG	General	$\Theta(\log(nw/\varepsilon)$
[Nis92, INW94]	PRG	General	$O(\log n \log(nw/\varepsilon))$
[BRRY10]	PRG	Regular	$\tilde{O}(\log n \log(w/\varepsilon))$
[KNP11, De11, Ste12]	PRG	Permutation	$O(\log n(\operatorname{poly}(w) + \log(1/\varepsilon))$
[BCG18, CL20], Thm 4.1	PRPG	General	$\tilde{O}(\log n \log nw + \log(1/\varepsilon))$
[HPV21]	PRG	Permutation (1 accept)	$\tilde{\Theta}(\log n \log(1/\varepsilon))$
Non-explicit [HPV21]	HSG	Permutation (1 accept)	$O(\log(n/\varepsilon))$
Theorem 1.4	PRPG	Permutation (1 accept)	$\tilde{O}(\log n\sqrt{\log(n/\varepsilon)} + \log(1/\varepsilon))$
Corollary 1.5	PRPG	Permutation	$\tilde{O}(\log n\sqrt{\log(nw/\varepsilon)} + \log(w/\varepsilon))$

2 Overview of Proofs

The starting point for our results are the recent space-efficient algorithms for estimating random-walk probabilities in directed graphs by Ahmadenijad, Kelner, Murtagh, Peebles, Sidford, and Vadhan [AKM+20], which are based on spectral graph theory and space-efficient Laplacian solvers. We interpret these algorithms as giving PRPGs with large seed length, which we then derandomize to obtain our results.

The specific problem considered by Ahmadenijad 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 estimate the acceptance probability of an ordered branching program as follows:

Definition 2.1 (Graph Associated with a Branching Program)**.** 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}, we can construct an associated layered graph $\mathcal{G}$ 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 Ahmadenijad et al. to the graph $\mathcal{G}$ with $s = (0, v_0)$, $t = (n, v_{\text{acc}})$, and $k = n$, we obtain an estimate of the acceptance probability of B to within $\pm\varepsilon$, just like an ε-PRPG for B would allow us to obtain. But a PRPG (G, ρ) is much more constrained than an arbitrary space-efficient algorithm, which can directly inspect the graph. Instead, a PRPG 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 $B(G(x_1)), \ldots, B(G(x_n))$ 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 (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 Ahmadenijad 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 PRPG based on that algorithm. The resulting PRPG matches the parameters of the PRPG of Braverman, Cohen, and Garg [BCG18], but has a significantly simpler proof (and is also simpler than the construction of Chatthopadhyay and Liao [CL20]). A similar construction was independently discovered by Cohen, Doron, Renard, Sberlo, and Ta-Shma (personal communication, January 2021).

2.1 PRPG 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 Ahmadenijad 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.)

Ahmadenijad 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

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

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 $\mathbf{L}^{-1}$.

Ahmadenijad et al. estimate $\mathbf{L}^{-1}$ in two steps. First, they observe that the Saks–Zhou derandomization of logspace [SZ99] can be used to produce, in deterministic space $O(\log(nw)\sqrt{\log(n)})$, approximations $\widetilde{\mathbf{B}_{i\dots j}}$ of the blocks $\mathbf{B}_{i\dots j}$ to within entrywise error $1/\operatorname{poly}(nw)$, resulting in an approximate pseudoinverse

$$\widetilde{\mathbf{L}^{-1}} = \begin{bmatrix} \widetilde{\mathbf{B}_{0\dots 0}} & \widetilde{\mathbf{B}_{0\dots 1}} & \widetilde{\mathbf{B}_{0\dots 2}} & \cdots & \widetilde{\mathbf{B}_{0\dots n}} \\ 0 & \widetilde{\mathbf{B}_{1\dots 1}} & \widetilde{\mathbf{B}_{1\dots 2}} & \cdots & \widetilde{\mathbf{B}_{1\dots n}} \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \ddots & \widetilde{\mathbf{B}_{n-1\dots n}} \\ 0 & 0 & 0 & \cdots & \widetilde{\mathbf{B}_{n\dots n}} \end{bmatrix}, \tag{1}$$

with the property that

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

where $\|\cdot\|_\infty$ denotes the ℓ_∞ operator norm, ie $||\mathbf{M}||_\infty = \sup_{x\neq 0} ||\mathbf{M}x||_\infty / ||x||_\infty$.

Next, Ahmadenijad 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, [AKM+20] Lemma 6.2)**.** *Let $||\cdot||$ be a submultiplicative norm on $N \times N$ real matrices. Given matrices $\mathbf{A}, \mathbf{E} \in \mathbb{R}^{N\times N}$ such that $||\mathbf{I}_N - \mathbf{E}\mathbf{A}|| \le \alpha$ for some constant $\alpha > 0$, let $\mathbf{P}_m = \sum_{i=0}^m (\mathbf{I}_N - \mathbf{E}\mathbf{A})^i\mathbf{E}$. Then $||\mathbf{I}_N - \mathbf{P}_m\mathbf{A}|| \le \alpha^{m+1}$.*

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

$$\widetilde{\mathbf{L}_\varepsilon} = \sum_{i=0}^{m} (\mathbf{I}_N - \widetilde{\mathbf{L}^{-1}}\mathbf{L})^i\mathbf{L} \tag{2}$$

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 Ahmadenijad et al.'s space bound of

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

Now we show how, with appropriate an modification, we can interpret this algorithm of Ahmadenijad et al. as a PRPG (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 $\widetilde{\mathbf{B}_{i\dots j}}$ to be the matrix whose (u,v)'th entry is the probability that, if we start at state u in the 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 $\widetilde{\mathbf{B}_{i\dots j}}$ to approximate $\mathbf{B}_{i\dots j}$ to within error $\pm 1/\operatorname{poly}(nw)$ as above, Nisan's pseudorandom generator requires seed length

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

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

Next, we observe from Equation 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 $\widetilde{\mathbf{B}_{i\dots j}}$. 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:

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

$$\pm \widetilde{\mathbf{B}_{i_0\cdots i_1}}\widetilde{\mathbf{B}_{i_1\cdots i_2}}\cdots\widetilde{\mathbf{B}_{i_{r-1}\cdots i_r}}, \tag{3}$$

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 (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 (3) equals the acceptance probability of the branching program on such a pseudorandom walk. Thus the algorithm now has the form required of a PRPG.

The seed length for the PRPG is the sum of the seed length s_{sum} needed to select a random term in the sum (using the coefficients of the PRPG to rescale the sum into a 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

$$s_{\text{sum}} = \log n^{O(m)} = O(m\log(n)) = O(\log_{nw}(1/\varepsilon)\log(n)) = O(\log(1/\varepsilon)).$$

The seed length needed for an individual term is at most

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

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 (3). Instead of using r independent seeds, we use dependent seeds generated using the Impagliazzo–Nisan–Wigderson pseudorandom generator [INW94]. 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 PRPG error of $O(\varepsilon)$, we should set $\gamma = \varepsilon/n^{O(m)} = 1/\varepsilon^{O(1)}$. Recalling that $r \le 2m+1 = O(\log_{nw}(1/\varepsilon))$, we attain a seed length of

$$\begin{aligned} s_{\text{sum}} + s'_{\text{term}} &= O(\log(1/\varepsilon)) + O(\log^2 nw) + O(\log\log_{nw}(1/\varepsilon)\cdot\log(1/\varepsilon)) \\ &= O(\log^2 nw + \log(1/\varepsilon)\cdot\log\log_{nw}(1/\varepsilon)),. \end{aligned}$$

which slightly improves over the bound of Braverman, Cohen, and Garg [BCG18], and is incomparable to that of Chattopadhyay and Liao [CL20]. Specifically, our first term of $O(\log(n)\log(nw))$ is better than [CL20] by a factor of $\log\log(nw)$, but our second term of $O(\log(1/\varepsilon)\log\log_{nw}(1/\varepsilon))$ is worse by a factor of $\log\log_{nw}(1/\varepsilon)$.

2.2 PRPG for Permutation Branching Programs

Now we give an overview of our PRPG for permutation branching programs, as stated in Theorem 1.4. This is based on the the algorithm of Ahmademnijad 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 Ahmadenijad 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 $\mathbf{W}$:

$$\mathbf{W}_0 = \begin{bmatrix} 0 & \mathbf{B}_1 & 0 & \cdots & 0 \\ 0 & 0 & \mathbf{B}_2 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \ddots & \mathbf{B}_n \\ \mathbf{J}_w & 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{4}$$

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 than having 0 in the lower-left block as we had before. Indeed, the powers of $\mathbf{W}$ look as follows:

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

where

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

Notice in particular that $\mathbf{W}_0^{n+1}$ will be a block-diagonal matrix with $\mathbf{J}_w$'s 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 [AKM+20], they instead estimate the Moore-Penrose pseudoinverse $\mathbf{I}_{(n+1)w} - \mathbf{W}_0$. 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 $c^n \geq 1/2$.

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 $\alpha < 1$ 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 Ahmadenijad 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:

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

Thus,

$$\begin{aligned} \left(\mathbf{I}_{2^q N} - c^{n+1}\mathbf{C}_q^{n+1} \otimes (\mathbf{I}_{n+1} \otimes \mathbf{J}_w)\right)\mathbf{L}^{-1} &= \mathbf{I}_{2^q N} + c\mathbf{C}_q \otimes \mathbf{W}_0 + c^2\mathbf{C}_q^2 \otimes \mathbf{W}_0^2 + \cdots c^n \mathbf{C}_q^n \otimes \mathbf{W}_0^n \\ &= \begin{bmatrix} \mathbf{I}_N & c\mathbf{W}_0 & c^2\mathbf{W}_0^2 & \cdots & c^n\mathbf{W}_0^n \\ c^n\mathbf{W}_0^n & \mathbf{I}_N & c\mathbf{W}_0 & \cdots & c^{n-1}\mathbf{W}_0^{n-1} \\ \vdots & & \ddots & & \vdots \\ c^2\mathbf{W}_0^2 & c^3\mathbf{W}_0^3 & c^4\mathbf{W}_0^4 & \ddots & c\mathbf{W}_0 \\ c\mathbf{W}_0^1 & c^2\mathbf{W}_0^2 & c^3\mathbf{W}_0^3 & \cdots & \mathbf{I}_N \end{bmatrix} \end{aligned}$$

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 $\mathbf{L} = \mathbf{I}_{2^q N} - c\mathbf{C}_q \otimes \mathbf{W}_0$, Ahmadenijad 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 Ahmadenijad et al. use the *derandomized square* [RV05] which allows for computing a sequence $\mathbf{W}_0, \mathbf{W}_1, \ldots, \mathbf{W}_q$ where $\mathbf{W}_i$ a sparsification of $\mathbf{W}_{i-1}^2$ with the property that $\mathbf{W}_q$ 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. They also 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 repeated derandomized squaring in the recursion, Ahmadenijad et al. obtain an approximate inverse $\widetilde{\mathbf{L}^{-1}}$ with the properties that:

1. The $N \times N$ blocks of $\widetilde{\mathbf{L}^{-1}}$ are each of the form $M\mathbf{W}_{i_1}\mathbf{W}_{i_2}\cdots\mathbf{W}_{i_r}$ where M is an easily computable matrix that does not depend on the branching program and $r = O(q)$
2. There is a submultiplicative matrix norm $\|\cdot\|_{\mathbf{F}}$ such that $\|\mathbf{I}_{2^qN} - \widetilde{\mathbf{L}^{-1}}\mathbf{L}\|_{\mathbf{F}} = O(q\delta)$. Moreover, δ approximation of $\mathbf{L}^+$ in $\mathbf{F}$ norm implies $\mathrm{poly}(n) \cdot \delta$ approximation in max norm.

Ahmadenijad et al. actually obtain that approximation in $\mathbf{F}$ norm implies $\mathrm{poly}(nw) \cdot \delta$ 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 $\widetilde{\mathbf{L}^{-1}}$ from $\mathbf{W}_0, \mathbf{W}_1, \ldots, \mathbf{W}_q$ in space

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

Item 2 allows obtaining entrywise error at most ε using preconditioned Richardson iterations (Lemma 2.2) with degree $m = O(\log(n/\varepsilon)/\log(1/q\delta))$ provided $\delta < 1/O(q)$, which has an additive space cost of:

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

Taking $\delta = 1/O(q)$ 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 PRPG 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 [RV05, HPV21]. Using this correspondence, the matrix $\mathbf{W}_i$ has the same structure as $\mathbf{W}^{2^i}$ (see Equation 4), 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 [HPV21], 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 Ahmadenijad 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 $\widetilde{\mathbf{B}_{(j-1)\ldots j}} = \mathbf{B}_j$ for $j = 1, \ldots, n$, we have the following analogue of Observation 2.3:

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

$$\pm \widetilde{\mathbf{B}_{i_0\cdots i_1}}\widetilde{\mathbf{B}_{i_1\cdots i_2}}\cdots\widetilde{\mathbf{B}_{i_{r-1}\cdots i_r}}, \tag{5}$$

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

As Subsection 2.1, the algorithm now has the form required of a PRPG and our only remaining challenge is to keep the seed length small. The seed length for the PRPG is the sum of the seed length needed to select a random term in the sum (using the coefficients of the PRPG to rescale the sum into a 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

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

The seed length needed for an individual term is at most

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

which again would be too expensive for us. To derandomize the product in Equation (5), we again use the INW generator, but rely on the analysis in [HPV21] for permutation branching programs to maintain a seed length that is indepndent 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

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

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

All in all, we attain a seed length of

$$\begin{aligned} s_{\text{sum}} + s'_{\text{term}} &\leq 2s'_{\text{term}} \\ &= 2s_{\text{INW}} + O((\log qm) \cdot \log(\log(qm)/\gamma)) \\ &= O(q \log(q/\delta)) + \tilde{O}(m \log n) + O(\log qm \cdot \log(n/\varepsilon)) \\ &= \tilde{O}\left(\log n \cdot \log(1/\delta) + \frac{\log(n/\varepsilon)}{\log(1/(\delta \log n))} \cdot \log n + \log\log n \cdot \log(n/\varepsilon)\right) \end{aligned}$$

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

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

Note that the choice of δ here is much smaller than in the Ahmadenijad et al. algorithm, which used $\delta = 1/\operatorname{polylog}(n)$. 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 Ahmadenijad et al.

2.3 Perspective

Some intuition for the ability of PRPGs to beat the parameters of PRGs can come from the study of *samplers* [Gol97]. A *sampler* for a class $\mathcal{F}$ of functions $f : \{0,1\}^m \to \mathbb{R}$ is 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 works 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 [Bla18, Agr19]. 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 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. $(f(x_1) + \cdots + f(x_S))/S$.

A PRG $G : \{0,1\}^s \to \{0,1\}^m$ for a class $\mathcal{F}$ can then be seen as a deterministic averaging sampler for the class $\mathcal{F}$, with randomness complexity and failure probability zero, and sample complexity $S = 2^s$. Indeed, the sampler simply outputs the set of all $S = 2^s$ outputs of G. Then we can see a PRPG 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. PRPGs 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 simple 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 $\Theta(\min\{1/\varepsilon^2\delta, 2^m\})$. (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.

This example suggests two areas of investigation for future work. 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)? Second, is there a benefit in the study of samplers in restricting attention to ones that output linear combinations like PRPGs? Perhaps these still retains some of the useful composition properties and connections to other pseudorandom objects that are enjoyed by averaging samplers (cf. [Zuc97, Vad12, Agr19]), while allowing for gains in sample and/or randomness complexity.

2.4 Organization of the Remaining Sections

In Section 3 we introduce arithmetic over pseudodistributions and recall properties of unit circle approximation. We then recall the Nisan [Nis92] generator, and the analysis of the INW generator from [HPV21]. 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 [AKM$^+$20] and [CKP$^+$17] and the analysis of the INW PRG from [HPV21].

3 Preliminaries

We first introduce basic operations on pseudodistributions, connections between PRGs for permutation branching programs and operations on graphs, and existing families of generators and expanders.

Following [RSV13], we will view branching programs not as boolean functions, but as a matrix valued function $\mathbf{B} : \{0,1\}^n \to \mathbb{R}^{w \times w}$ where $\mathbf{B}[s]_{i,j} = 1$ if the branching program started at state i ends at state j upon reading s. In all cases, we will index matrices from *zero*, to correspond with the conventional notation for labeling layers of branching programs.

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 where $\mathbf{B}_t(s)_{i,j} = 1$ if $B_t(i,s) = j$ and 0 otherwise. For $0 \leq i < j \leq n$ let $\mathbf{B}_{i..j}$ be defined as $\mathbf{B}_{i..j}(s_{i+1} \ldots s_j) = \mathbf{B}_{i+1}(s_{i+1}) \ldots \mathbf{B}_j(s_j)$, and let $\mathbf{B} = \mathbf{B}_{0..n}$.

We can define the matrix form of pseudodistribution generators. We write $U_n = U_{\{0,1\}^n}$ for convenience.

Definition 3.2. Let (X, Y) be joint random variables on $\{0,1\}^n \times \mathbb{R}$ and $||\cdot||$ a norm on $w \times w$ matrices. We say that (X, Y) is an **ε-pseudorandom pseudodistribution** for a class $\mathcal{B}$ of length n, width w ordered branching programs if for all $\mathbf{B} \in \mathcal{B}$ we have

$$||\mathbb{E}[Y \cdot \mathbf{B}[X]] - \mathbb{E}[\mathbf{B}[U_n]]|| < \varepsilon.$$

We say that $(G, \rho) : \{0,1\}^s \to \{0,1\}^n \times \mathbb{R}$ is an **ε-pseudodistribution generator** (ε-PRPG) for a class $\mathcal{B}$ if $(G(U_s), \rho(U_s))$ is an ε-pseudodistribution for $\mathcal{B}$. We write $\overline{\mathbf{B}}[(Y, X)]$ to denote $\mathbb{E}[Y \cdot \mathbf{B}[X]]$.

In contrast to pseudorandom generators, we can define rules for "arithmetic" on PRPGs, which naturally translate into operations on matrix forms.

Definition 3.3 (Sum Rule for PRPGs)**.** Given PRPGs $F_a = (G_a, \rho), F_b = (G_b, \rho_b)$ with seed length s, let $F_a + F_b$ be the PRPG with seed length $s + 1$, where for $(x, b) \in \{0,1\}^s \times \{0,1\}$ we define

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

Remark 3.4. For any branching program $\mathbf{B}$ and PRPGs F_a, F_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.

Definition 3.5 (Product Rule for PRPGs)**.** Given PRPGs $F_a = (G_a, \rho_a), F_b = (G_b, \rho_b)$ seed length s and output lengths n, n' respectively, let $F_a F_b$ be the PRPG with seed length $2s$ and output length $n + n'$, where for $(x, y) \in \{0,1\}^s \times \{0,1\}^s$ we define

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

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

Remark 3.6. For any branching programs $\mathbf{B}, \mathbf{B}'$ of length n, n' and equal width and PRPGs F_a, F_b of output length n, n',

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

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.

Slightly abusing notation, if F is an empty PRG on zero bits we define $\overline{\mathbf{B}}[F] = \mathbf{I}_w$. This is purely for cleanliness of later derivations. We also implicitly define the sum and concatenation of PRPGs with different seed lengths, by first padding the shorter seed to equal that of the longer.

3.1 Matrix Approximation Preliminaries

We introduce basic notation and unit-circle approximation, a measure of closeness of approximation for matrices that allows for error bounds independent of size.

- For a complex number $z \in \mathbb{C}$ we write z^* to denote the complex conjugate of z and $|z|$ to denote the magnitude of z.
- For a matrix $\mathbf{A} \in \mathbb{C}^{N\times N}$ we write $\mathbf{A}^*$ to denote its conjugate transpose and write $\mathbf{U_A} = (\mathbf{A} + \mathbf{A}^*)/2$ to denote its **symmetrization**.
- For $\mathbf{A} \in \mathbb{C}^{N\times N}$ we define $||\mathbf{A}||_\infty = \max_i \sum_{j=1}^n |\mathbf{A}_{i,j}|$.
- We say a Hermitian matrix $\mathbf{A}$ is **positive semidefinite** (PSD) or write $\mathbf{A} \succeq 0$ if $x^*\mathbf{A}x \geq 0$ for all $x \in \mathbb{C}^N$. For two Hermitian matrices $\mathbf{A}, \mathbf{E}$, we use $\mathbf{A} \succeq \mathbf{E}$ to denote $\mathbf{A} - \mathbf{E} \succeq 0$ and define $\preceq$ analogously.
- For any vector norm $||\cdot||$ defined on $\mathbb{C}^n$ we define the matrix semi-norm on $\mathbb{C}^{N\times N}$ by $||\mathbf{A}|| = \max_{x\neq 0} ||\mathbf{A}x||/||x||$. For a PSD matrix $\mathbf{H}$ and vector x we have $||x||_\mathbf{H} = \sqrt{x^*\mathbf{H}x}$ and the matrix seminorm is defined accordingly.

We give a basic proposition to be used for condition number bounds.

Proposition 3.7. *Let $\mathbf{H} \in \mathbb{C}^{N\times N}$ be a Hermitian positive definite matrix with minimum eigenvalue α and maximum eigenvalue β. Then for any $x \in \mathbb{C}^N$, $\alpha||x||_2^2 \leq ||x||_\mathbf{H}^2 \leq \beta||x||_2^2$.*

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

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

and the lower bound is nearly identical. □

We next introduce a notion of approximation that is preserved under cycle lifting and other essential operations.

Definition 3.8 (Unit-Circle Approximation [AKM$^+$20])**.** For $\mathbf{A}, \mathbf{E} \in \mathbb{C}^{N\times N}$ and $\varepsilon \in (0, 1/2)$, we say $\mathbf{A}$ is an ε-**unit-circle approximation** of $\mathbf{E}$, denoted $\mathbf{A} \overset{\circ}{\approx}_\varepsilon \mathbf{E}$, if

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

Including the magnitude operation in the right hand side forces the approximation to be exact for all eigenspaces with eigenvalues of complex magnitude 1, and this property is essential for the preservation of approximation under high powers. The unit-circle approximation is developed in [AKM$^+$20].

Proposition 3.9. *For $\mathbf{A}, \mathbf{E} \in \mathbb{C}^{N\times N}$, if $\mathbf{A} \overset{\circ}{\approx}_\varepsilon \mathbf{E}$ and $c \in \mathbb{C}$ with $|c| < 1$ then $c\mathbf{A} \overset{\circ}{\approx}_\varepsilon c\mathbf{E}$.*

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

$$|x^*(c\mathbf{A} - c\mathbf{E})y| \leq \frac{\varepsilon}{2}(||x||^2 + ||y||^2 - |x^*\mathbf{E}x + y^*\mathbf{E}y|) \leq \frac{\varepsilon}{2}(||x||^2 + ||y||^2 - |x^*c\mathbf{E}x + y^*c\mathbf{E}y|). \quad \square$$

Proposition 3.10. *Let* $\mathbf{A}, \mathbf{E}, \mathbf{X}, \mathbf{Y} \in \mathbb{R}^{N \times N}$ *where* $\mathbf{A} \overset{\circ}{\approx}_\varepsilon \mathbf{X}$ *and* $\mathbf{E} \overset{\circ}{\approx}_\varepsilon \mathbf{Y}$*. Then for any* $\lambda \in [0,1]$*,* $\lambda\mathbf{A} + (1-\lambda)\mathbf{E} \overset{\circ}{\approx}_\varepsilon \lambda\mathbf{X} + (1-\lambda)\mathbf{Y}$

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

$$\begin{aligned}
|x^*(\lambda\mathbf{A} + (1-\lambda)\mathbf{E} - (\lambda\mathbf{X} + (1-\lambda)\mathbf{Y}))y| &\leq \lambda|x^*(\mathbf{A}-\mathbf{X})y| + (1-\lambda)|x^*(\mathbf{E}-\mathbf{Y})y| \\
&\leq \frac{\varepsilon}{2}(||x||_2^2 + ||y||_2^2 - \lambda|x^*\mathbf{X}x + y^*\mathbf{X}y| - (1-\lambda)|x^*\mathbf{Y}x + y^*\mathbf{Y}y|) \\
&\leq \frac{\varepsilon}{2}(||x||_2^2 + ||y||_2^2 - |x^*(\lambda\mathbf{X} + (1-\lambda)\mathbf{Y})x + y^*(\lambda\mathbf{X} + (1-\lambda)\mathbf{Y})y|)
\end{aligned}$$

□

3.2 Explicit Primitive Generators

To prove Theorem 4.1 we make use of the existing pseudorandom generator of Nisan [Nis92].

Theorem 3.11 ([Nis92])**.** *Given* n, w *and* $\varepsilon \in (0, 1/2)$*, there exists an explicit pseudorandom generator* $G : \{0,1\}^s \to \{0,1\}^n$ *such that for any branching program* $\mathbf{B}$ *of length* n *and width* w*,*

$$\left|\left|\overline{\mathbf{B}}[U_n] - \overline{\mathbf{B}}[G]\right|\right|_{\max} \leq \varepsilon,$$

and $s = O(\log(n)\log(nw/\varepsilon))$*.*

To prove Theorem 1.4 we make use of a consequence of [HPV21]. To do so, we require a notion of distributions over graphs.

Definition 3.12. Given a length n permutation branching program B with transition functions $B_1, \ldots, B_n, B_0$ where we require a transition function from layer n to layer 0, define the **transition graph** of B as a function $\hat{\mathbf{B}} : \{0,1\} \to \mathbb{R}^{(n+1)w \times (n+1)w}$ where

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

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

We then state a convenient form of the theorem.

Theorem 3.13 (Consequence of [HPV21] Theorem 1.4)**.** *For all* $\ell \in \mathbb{N}$ *and* $\varepsilon \in (0, 1/2)$*, there is a family of explicit PRGs* $\mathrm{INW}_0, \ldots, \mathrm{INW}_\ell$ *such that* $\mathrm{INW}_0 : \{0,1\} \to \{0,1\}$ *is the identity map on one bit, and for all* $i \in [\ell]$*,* INW_i *has seed length* $b_i = O(i\log(1/\varepsilon))$*. Furthermore, for all* $i \in \{0, \ldots, \ell-1\}$*, for any permutation branching program* B *with transition graph* $\hat{\mathbf{B}}$*,*

$$\mathbb{E}[\hat{\mathbf{B}}^{(2^{i+1})}[\mathrm{INW}_{i+1}(U_{b_{i+1}})]] \overset{\circ}{\approx}_\varepsilon \mathbb{E}[\hat{\mathbf{B}}^{(2^{i+1})}[\mathrm{INW}_i(U_{b_i}), \mathrm{INW}_i(U_{b_i})]].$$

4 Pseudodistributions for General Branching Programs

We first develop the PRPG for general branching programs, as it outlines the some of the ideas and constructions used later but with simpler analysis.

Theorem 4.1. *For all n, w and $\varepsilon \in (0, 1/2)$, there is an explicit* $\mathrm{poly}(1/\varepsilon)$ *bounded pseudorandom pseudodistribution generator* GEN *with seed length $O(\log(n)\log(nw) + \log(1/\varepsilon)\log\log_{nw}(1/\varepsilon))$ such that for every (n, w) branching program $\mathbf{B}$,*

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

4.1 Richardson Iteration for PRG Error Reduction

We first define a set of PRPGs of all lengths for a given error parameter. We write $R : \{0,1\} \to \{0,1\}$ to mean the trivial PRG on one bit.

Definition 4.2. Given $n, w \in \mathbb{N}$, for all $r \in [n]$ let N_r be the explicit PRG obtained from Theorem 3.11 with width w, length r and error $\delta = 1/4n^2w^2$. Then for all $r \in [n]$ define the PRPG

$$D_r = N_{r-1}R - N_r.$$

We then define a set of products of such PRPGs. The index set is equivalent to all possible divisions of the layers $\{0, \ldots, k\}$ for $k \leq n$ into at most ℓ sections.

Definition 4.3. Given $\ell, n \in \mathbb{N}$, define the index set $\mathbb{V}_{n,\ell}$ as

$$\mathbb{V}_{n,\ell} = \{(d_1, \ldots d_k) : d_i \in \mathbb{Z}^+, \quad 0 \leq k \leq \ell, \quad \sum_i d_i \leq n\}.$$

For $\sigma \in \mathbb{V}_{n,\ell}$ we write $|\sigma|$ to denote the k corresponding to this σ. Note that this includes the empty tuple $\sigma = ()$. Now given $w \in \mathbb{N}$, let D_r be defined as in Definition 4.2 with $n = n, w = w$. For all $\sigma = (d_1, \ldots, d_k) \in \mathbb{V}_{n,\ell}$, let $g = n - \sum_{i=1}^{k} d_i$ and define $M_\sigma = [\prod_{i=1}^{k} D_{d_i}]N_g$.

We prove the sum over all such PRPGs is a good approximation of truly random input.

Lemma 4.4. *Fix arbitrary $n, w \in \mathbb{N}$ and $\varepsilon \in (0, 1/2)$. Define $\ell = \lceil \log_{nw}(1/\varepsilon) \rceil + 1$ and let $\mathbb{V}_{n,\ell}$ and M_σ be defined as in Definition 4.3 with $n = n, w = w$ and $\ell = \ell$. Then for any (n, w) branching program $\mathbf{B}$,*

$$\left|\left| \sum_{\sigma \in \mathbb{V}_{n,\ell}} \overline{\mathbf{B}}[M_\sigma] - \overline{\mathbf{B}}[U_n] \right|\right|_{\max} \leq \varepsilon/2.$$

We remark that each M_σ is a sum of 2^ℓ products of at most $2\ell + 1$ explicit PRGs, each with seed length $O(\log(nw)\log(n))$, multiplied by a sign. We prove the lemma by relating the summation to the output of preconditioned richardson iteration applied to a matrix chosen to have a desired inverse.

Proposition 4.5. *Fix $n, w \in \mathbb{N}$. Let N_r be as defined in Definition 4.2 with $n = n, w = w$. Then for any (n, w) branching program $\mathbf{B}$, define the path lift* ***Laplacian*** *$\mathbf{L} \in \mathbb{R}^{(n+1)w \times (n+1)w}$ of $\mathbf{B}$ as the $(n+1) \times (n+1)$ block matrix where*

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

with inverse

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

Then define

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

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

To prove this, we describe the structure of **Err**.

Lemma 4.6. *Fix* $n, w \in \mathbb{N}$ *and let* D_r *be defined as in Definition 4.2. The error matrix* **Err** *as defined in Proposition 4.5 has form:*

$$\mathbf{Err}_{i,j} = \begin{cases} \overline{\mathbf{B}}_{i..j}[D_{j-i}] & i < j \\ 0 & i \geq j \end{cases}.$$

Proof. We first consider when $i < j$. From the structure of L we have

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

And analyzing the rest of the cases is immediate, and then $(\mathbf{Err})_{i,j}$ matches the desired form. □

We can then prove the proposition.

Proof Of Proposition 4.5. We bound the ∞ norm of each block of **Err** then take a union bound over the at most n nonzero blocks in each row. Diagonal and lower triangular blocks are trivially 0, and then for any $i < j$ we have

$$\begin{aligned} ||\mathbf{Err}_{i,j}||_\infty &= ||\overline{\mathbf{B}}_{i..j}[D_{j-i}]||_\infty \\ &\leq ||\overline{\mathbf{B}}_{i..j}[N_{j-i-1}R] - \overline{\mathbf{B}}_{i..j}[R^{j-i}] + \overline{\mathbf{B}}_{i..j}[R^{j-i}] - \overline{\mathbf{B}}_{i..j}[N_{j-i}]||_\infty \\ &\leq ||\overline{\mathbf{B}}_{i..j-1}[N_{j-i-1}] - \overline{\mathbf{B}}_{i..j-1}[R^{j-i-1}]||_\infty ||\overline{\mathbf{B}}_{j-1..j}[R]||_\infty + ||\overline{\mathbf{B}}_{i..j}[R^{j-i}] - \overline{\mathbf{B}}_{i..j}[N_{j-i}]||_\infty \\ &\leq \frac{1}{4n^2w} + \frac{1}{4n^2w}. \end{aligned}$$

Where the first equality comes from Lemma 4.6, and the final from Theorem 3.11 where we lose a factor of w from going from max to ∞ norm. Thus a union bound over all nonzero j of $\mathbf{Err}_{i,\cdot}$ for all i completes the proof. □

Thus, by replacing truly random input with a PRG of the correct length, we obtain a weak approximation of $\mathbf{L}^{-1}$. We then use preconditioned Richardson iteration to boost this to a high quality approximation, and by describing this output in terms of a PRPG prove that this generator approximates truly random input for all branching programs.

Lemma 2.2 (Preconditioned richardson iteration, [AKM+20] Lemma 6.2)**.** *Let $||\cdot||$ be a submultiplicative norm on $N \times N$ real matrices. Given matrices $\mathbf{A}, \mathbf{E} \in \mathbb{R}^{N \times N}$ such that $||\mathbf{I}_N - \mathbf{EA}|| \leq \alpha$ for some constant $\alpha > 0$, let $\mathbf{P}_m = \sum_{i=0}^{m} (\mathbf{I}_N - \mathbf{EA})^i \mathbf{E}$. Then $||\mathbf{I}_N - \mathbf{P}_m \mathbf{A}|| \leq \alpha^{m+1}$.*

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

We can then apply Richardson iteration to boost our weak estimate of $\mathbf{L}^{-1}$ to a strong estimate.

Lemma 4.7. *Fix $n, w \in \mathbb{N}$ and $\varepsilon \in (0, 1/2)$ and let D_r be defined as in Definition 4.2. Set $\ell = \lceil \log_{nw}(1/\varepsilon) \rceil + 1$. Then for any (n, w) branching program $\mathbf{B}$, let $\mathbf{L}, \widetilde{\mathbf{L}^{-1}}$ and $\mathbf{Err}$ be defined as in Proposition 4.5. Then*

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

Proof. We apply Lemma 2.2 with $\mathbf{A} = \mathbf{L}$, $\mathbf{E} = \widetilde{\mathbf{L}^{-1}}$, $||\cdot|| = ||\cdot||_\infty$ and $\alpha \leq 1/2nw$ (which follows from Proposition 4.5) and obtain $\mathbf{P}_m = \sum_{i=0}^{\ell} \mathbf{Err}^i \cdot \widetilde{\mathbf{L}^{-1}}$ satisfying $||\mathbf{I} - \mathbf{P}_m \mathbf{L}||_\infty \leq \varepsilon/2nw$. Finally,

$$\begin{aligned} ||\mathbf{P}_m - \mathbf{L}^{-1}||_{\max} &\leq ||(\mathbf{I} - \mathbf{P}_m \mathbf{L})\mathbf{L}^{-1}||_\infty \\ &\leq ||\mathbf{I} - \mathbf{P}_m \mathbf{L}||_\infty ||\mathbf{L}^{-1}||_\infty \\ &\leq \frac{\varepsilon}{2nw}(n+1) \end{aligned}$$ □

Given this error guarantee, it remains to interpret the "output" of Richardson iteration in an oblivious manner. Intuitively, taking powers of the **Err** matrix corresponds to concatenating PRPGs to create more complex pseudodistributions on layers.

Lemma 4.8. *Fix $n, w, \ell \in \mathbb{N}$ and $\varepsilon \in (0, 1/2)$. Let $\{M_\sigma : \sigma \in \mathbb{V}_{n,\ell}\}$ be as defined in Definition 4.3. For any (n, w) branching program $\mathbf{B}$, let $\mathbf{Err}, \widetilde{\mathbf{L}^{-1}}$ be as defined in Proposition 4.5. Then*

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

Proof. Fix any $0 \le r \le \ell$.

$$\begin{aligned}
(\mathbf{Err}^r \cdot \widetilde{\mathbf{L}^{-1}})_{0,n} &= \sum_{l_i \in \{0..n\}^r} \mathbf{Err}_{0,l_1} (\prod_{i=1}^{r-1} \mathbf{Err}_{l_i,l_{i+1}}) \widetilde{\mathbf{L}^{-1}}_{l_r,n} \\
&= \sum_{0<l_1<\ldots<l_r\le n} \mathbf{Err}_{0,l_1} (\prod_{i=1}^{r-1} \mathbf{Err}_{l_i,l_{i+1}}) \widetilde{\mathbf{L}^{-1}}_{l_r,n} \\
&= \sum_{0<l_1<\ldots<l_r\le n} \overline{\mathbf{B}}[D_{l_1} (\prod_{i=1}^{r-1} D_{l_{i+1}-l_i}) N_{n-l_r}] \\
&= \sum_{(d_i)_{i\in[r]}: d_i\in\mathbb{N}, g=\sum_{i=1}^r d_i \le n} \overline{\mathbf{B}}[(\prod_{i=1}^{r} D_{d_i}) N_{n-g}] \\
&= \sum_{\sigma\in\mathbb{V}_{n,\ell}, |\sigma|=r} \overline{\mathbf{B}}[M_\sigma]
\end{aligned}$$

where the second and third lines follow from Lemma 4.6. Thus,

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

We can then prove the main lemma.

Proof of Lemma 4.4. Fix $n, w \in \mathbb{N}$ and $\varepsilon \in (0, 1/2)$ and let $\ell = \lceil \log_{nw}(1/\varepsilon) \rceil + 1$. Fix an arbitrary (n, w) branching program $\mathbf{B}$. Let $\mathbf{L}, \widetilde{\mathbf{L}^{-1}}$ and $\mathbf{Err}$ be defined as in Proposition 4.5. Then

$$\begin{aligned}
\varepsilon/2 &\ge \left|\left| (\sum_{r=0}^{\ell} \mathbf{Err}^i \cdot \widetilde{\mathbf{L}^{-1}})_{0,n} - (\mathbf{L}^{-1})_{0,n} \right|\right|_{\max} \\
&= \left|\left| \sum_{\sigma\in\mathbb{V}_{n,\ell}} \overline{\mathbf{B}}[M_\sigma] - \overline{\mathbf{B}}[U_n] \right|\right|_{\max}
\end{aligned}$$

Where the first line follows from Lemma 4.7 and the second from Lemma 4.8. $\square$

Note that $|\mathbb{V}_{n,\ell}| \le n^{O(\ell)} = \mathrm{poly}(1/\varepsilon)$ and for all σ, M_σ is a 2^ℓ bounded explicit PRPG, so $\sum_{\sigma\in\mathbb{V}_{n,\ell}} M_\sigma$ is already an explicit, $\mathrm{poly}(1/\varepsilon)$ bounded ε-PRPG for (n, w) branching programs. Unfortunately, each PRPG M_σ has seed length $\Omega(\ell \log(n) \log(nw)) = \Omega(\log(n) \log(1/\varepsilon))$. However, we can then apply standard derandomization results and use that $\ell = O(\log_{nw}(1/\varepsilon))$ to replace these PRPGs with suitable approximations.

4.2 Approximation of PRPG Products

Lemma 4.9. *Given $w \in \mathbb{N}$ and $\delta \in (0, 1/2)$ and a tuple of explicit PRGs $M_1, \ldots, M_r$ where $M_i : \{0,1\}^s \to \{0,1\}^{l_i}$ and given i, M_i is constructible in space $O(s)$, there exists an explicit PRG $\widetilde{M}$ such that for any branching program $\mathbf{B}$ of width w and length $\sum_{i=1}^r l_i$, we have $||\overline{\mathbf{B}}[\widetilde{M}] - \overline{\mathbf{B}}[M_1 \ldots M_r]||_{\max} < \delta$ and $\widetilde{M}$ has seed length $O(s + \log(r) \log(rw/\delta))$.*

This result is a direct consequence of the derandomization lemma below. We use the formulation as stated in Lemma 11 of [CL20].

Lemma 4.10 (INW [INW94])**.** *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 any $\delta \in (0,1/2)$ 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 any (l_1, w) branching program $\mathbf{B}$ and (l_2, w) branching program $\mathbf{B}'$, we have*

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

Given Lemma 4.9, we can reduce the seed of all PRPGs appearing in the Richardson polynomial.

Corollary 4.11. *Fix $w \in \mathbb{N}$ and $\gamma \in (0,1/2)$ and a length n PRPG $M = \sum_{i=1}^{t} \tau_i \cdot G_{i_1} \dots G_{i_k}$ where $\tau_i \in \{-1, 1\}$ and for all i_j, G_{i_j} is an explicit PRG with seed length s, and given i, τ_i and the index set i_j are constructible in space $O(s)$. Then there is an explicit t bounded PRPG F with seed length $O(s + \log(wkt/\gamma)\log(k))$ such that for any (n, w) branching program $\mathbf{B}$,*

$$\left|\left|\overline{\mathbf{B}}[F] - \overline{\mathbf{B}}[M]\right|\right|_{\max} \leq \gamma.$$

Proof. For all $i \in [t]$, let F_i be the PRPG obtained from applying Lemma 4.9 to $G_{i_1} \dots G_{i_k}$ with $\delta = \gamma/t$. Then F_i is explicit and has seed length $O(s + \log(k)\log(wkt/\gamma))$.

Then defining $F = \sum_i \tau_i F_i$, we have for any branching program $\mathbf{B}$

$$\begin{aligned}\left|\left|\overline{\mathbf{B}}[F] - \overline{\mathbf{B}}[M]\right|\right|_{\max} &\leq \sum_{i=1}^{t} \left|\left|\overline{\mathbf{B}}[\tau_i F_i] - \overline{\mathbf{B}}[\tau_i G_{i_1} \dots G_{i_k}]\right|\right|_{\max} \\ &\leq \frac{\gamma}{t} t\end{aligned}$$

Then by Definition 3.3 F is t bounded and has seed length $O(s + \log(t) + \log(k)\log(wkt/\gamma))$. □

4.3 From Richardson Iterations to PRPGs

We are now prepared to prove Theorem 4.1.

Theorem 4.1. *For all n, w and $\varepsilon \in (0,1/2)$, there is an explicit $\mathrm{poly}(1/\varepsilon)$ bounded pseudorandom pseudodistribution generator GEN with seed length $O(\log(n)\log(nw) + \log(1/\varepsilon)\log\log_{nw}(1/\varepsilon))$ such that for every (n, w) branching program $\mathbf{B}$,*

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

Proof. Let $\ell = \lceil \log_{nw}(1/\varepsilon) \rceil + 1$ and let $\{M_\sigma : \sigma \in \mathbb{V}_{n,\ell}\}$ be defined as in Definition 4.3 with $\ell = \ell, n = n$ and $w = w$. Observe that $|\mathbb{V}_{n,\ell}| \leq n^{\log_{nw}(1/\varepsilon)} = \mathrm{poly}(1/\varepsilon)$.

For all $\sigma \in \mathbb{V}_{n,\ell}$, let GEN_σ be the explicit PRPG obtained from applying Corollary 4.11 with $M = M_\sigma$, $w = w$ and $\gamma = \varepsilon/2|\mathbb{V}_{n,\ell}|$. In the notation of the corollary, $k = O(\log_{nw}(1/\varepsilon))$ and $t = 1/\mathrm{poly}(\varepsilon)$. Thus GEN_σ is explicit, $\mathrm{poly}(1/\varepsilon)$ bounded, and has seed length $s_{\mathrm{term}} = O(\log(nw)\log(n) + \log(w/\varepsilon)\log\log_{nw}(1/\varepsilon))$. As we have $\log(n) = \Omega(\log\log_{nw}(1/\varepsilon))$ we can ignore the $\log(w)\log\log_{nw}(1/\varepsilon)$ term.

Finally, let $s_{\mathrm{sum}} = \lceil \log(|\mathbb{V}_{n,\ell}|) \rceil$ and define $\mathrm{GEN} : \{0,1\}^{s_{\mathrm{sum}}} \times \{0,1\}^{s_{\mathrm{term}}} \to \{0,1\}^n$ by

$$\mathrm{GEN} = \sum_{\sigma \in \mathbb{V}_{n,\ell}} \mathrm{GEN}_\sigma .$$

Now fix an arbitrary (n, w) branching program $\mathbf{B}$. We have

$$\begin{aligned}\left|\left|\overline{\mathbf{B}}[\mathrm{GEN}] - \overline{\mathbf{B}}[U_n]\right|\right|_{\max} &= \left|\left|\overline{\mathbf{B}}[\sum_{\sigma\in\mathbb{V}_{n,\ell}} \mathrm{GEN}_\sigma] - \overline{\mathbf{B}}[U_n]\right|\right|_{\max} \\ &\leq \sum_{\sigma\in\mathbb{V}_{n,\ell}} \left|\left|\overline{\mathbf{B}}[\mathrm{GEN}_\sigma] - \sum_{\sigma\in\mathbb{V}_{n,\ell}} \overline{\mathbf{B}}[M_\sigma]\right|\right|_{\max} + \left|\left|\sum_{\sigma\in\mathbb{V}_{n,\ell}} \overline{\mathbf{B}}[M_\sigma] - \overline{\mathbf{B}}[U_n]\right|\right|_{\max} \\ &\leq \frac{\varepsilon}{2|\mathbb{V}_{n,\ell}|}|\mathbb{V}_{n,\ell}| + \frac{\varepsilon}{2}\end{aligned}$$

where the final line comes from our choice of error in Corollary 4.11 and Lemma 4.4.

It remains to show seed length and explicitness. We have that GEN is explicit and $\mathrm{poly}(1/\varepsilon)$ bounded by Definition 3.3. The seed length is bounded as $s = s_{\mathrm{sum}} + s_{\mathrm{term}}$, and as

$$s_{\mathrm{sum}} = O(\ell \log(n)) = O(\log_{nw}(1/\varepsilon)\log(n)) = O(\log(1/\varepsilon))$$

and

$$s_{\mathrm{term}} = O(\log(n)\log(nw) + \log(1/\varepsilon)\log\log_{nw}(1/\varepsilon))$$

so we obtain the desired result. □

5 Pseudodistributions for Permutation Branching Programs

There are two major modifications of the above method for permutation branching programs. First, we use machinery from [AKM+20] for a more sophisticated estimate of the norm of the error matrix. Second, we use tools from [HPV21] to approximate concatenations of PRPGs independent of width.

5.1 Cycle Lift Correspondence

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-PRPG for permutation branching programs must also ε-fool permutation branching programs, as the final layers could be the identity.

We first give an analogue of Lemma 4.4, except that the base PRPGs are somewhat more complicated, so we defer their precise definition till later. We define our index set in an isomorphic but more convenient manner.

Definition 5.1. For all $n, \ell \in \mathbb{N}$ define the index set $\mathbb{V}_{n,\ell}$ as

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

For $\sigma \in \mathbb{V}_{n,\ell}$ we write $|\sigma|$ to denote the k corresponding to this σ.

We can then state an analogue of Lemma 4.4.

Theorem 5.2. *For all $n \in \mathbb{N}$ and $\varepsilon, \delta \in (0, 1/2)$, set $\ell = \lceil \log_{1/\delta}(1/\varepsilon)\rceil$. Then there exists a family $\{M_\sigma : \sigma \in \mathbb{V}_{n,\ell}\}$ of PRPGs, where for all σ, M_σ is a sum over 2^ℓ products of $O(\ell \log(n))$ explicit PRGs, each with seed length $O(\log(n)\log(\log(n)/\delta))$, multiplied by a sign. Furthermore, for any permutation branching program $\mathbf{B}$ of length n and arbitrary width,*

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

For the remainder of the subsection, for an arbitrary length n, width w permutation BP $\mathbf{B}$ let $N = (n+1)w$. Given the program $\mathbf{B}$, we define a graph that "holds" information about the (pseudo)-distribution of a function over the program.

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

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

Recall that for all i we define $\hat{\mathbf{B}}^{(i)} : \{0,1\}^i \to \mathbb{R}^{(n+1)w \times (n+1)w}$ as $\hat{\mathbf{B}}^{(i)}[s] = \hat{\mathbf{B}}[s_1]\hat{\mathbf{B}}[s_2]\dots\hat{\mathbf{B}}[s_i]$. Finally, given a PRPG $(G, \rho) : \{0,1\}^s \to \{0,1\}^m$ we abuse notation and define

$$\hat{\mathbf{B}}^{(m)}[(G,\rho)] = \mathop{\mathbb{E}}_{x \leftarrow U_s}[\rho(x)\hat{\mathbf{B}}^{(m)}[G(s)]].$$

We first prove a basic result on the structure of these matrices.

Proposition 5.4. *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:*

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

Proof. The only nonzero blocks of $\hat{\mathbf{B}}$ have index $(i, i+1 \mod n+1)$, and we implicitly mod the indices for notational convenience. Thus,

$$(\hat{\mathbf{B}}^{(x)}[s])_{j,k} = (\prod_{i=1}^{x} \hat{\mathbf{B}}[s_i]_{j+(i-1),j+i})_{j,k}$$

Clearly if $k \neq j + x \mod n+1$ this is zero. Otherwise, if $\hat{\mathbf{B}}[s_i]_{n,0} = \mathbf{J}_w$ appeared in this product, all other matrices in the product are doubly stochastic and so $(\hat{\mathbf{B}}^{(x)}[s])_{j,k} = \mathbf{J}_w$. Otherwise the product is of the form $\prod_{i=1}^{x} \mathbf{B}[s_i]_{j+(i-1),j+i} = \mathbf{B}_{j..j+x}[s]$ as desired. □

Remark 5.5. We have that $\hat{\mathbf{B}}^{(n+1)}[s] = \mathbf{I}_{n+1} \otimes \mathbf{J}_w$, independent of s. Furthermore,

$$(\mathbf{I}_{n+1} \otimes \mathbf{J}_w)\hat{\mathbf{B}}^{(x)}[s] = (\mathbf{I}_{n+1} \otimes \mathbf{J}_w)\hat{\mathbf{B}}^{(x)}[s'] = \hat{\mathbf{B}}^{(x)}[s'](\mathbf{I}_{n+1} \otimes \mathbf{J}_w)$$

for any x and s, s', which follows from the fact that $\mathbf{B}$ is a *permutation* branching program.

This will enable higher powers to exactly cancel in the error reduction procedure.

Definition 5.6. Given $n \in \mathbb{N}$ and $\delta \in (0, 1/2)$, let $\mathrm{INW}_0, \dots, \mathrm{INW}_q$ be the PRGs obtained from applying Theorem 3.13 with $n = 2^q$ and error $\delta = \delta/40q$. These generators have seed length $b_i = O(q \log(q/\delta))$. Then for any permutation BP B, let $\hat{\mathbf{B}}$ be defined as in Definition 5.3 and for all $i \in \{0,\dots,\ell\}$ define

$$\mathbf{W}_i = \mathbb{E}[\hat{\mathbf{B}}^{(2^i)}[\mathrm{INW}_i(U_{b_i})]].$$

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. Furthermore, these matrices are well structured, as products correspond to concatenations of PRG outputs.

Proposition 5.7. *For any sequence $\mathbf{W}_{i_1} \dots \mathbf{W}_{i_r}$ and $k, l \in \{0, \dots, n\}$, let $m = k + \sum_{j=1}^{r} 2^{i_j} \mod n+1$. Then*

$$(\prod_{j=1}^{r} \mathbf{W}_{i_j})_{k,l} = \begin{cases} 0 & k \neq m \\ \mathbf{J}_w & m \leq k \\ \overline{\mathbf{B}}_{k..l}[(\prod_{j=1}^{r} \mathrm{INW}_{i_j})] & k < m \end{cases}$$

Proof. We write the product as

$$\begin{aligned} \prod_{j=1}^{r} \mathbf{W}_{i_j} &= \prod_{j=1}^{r} \mathop{\mathbb{E}}_{x \leftarrow U_{b_{i_j}}} [\hat{\mathbf{B}}^{(2^{i_j})}[\mathrm{INW}_{i_j}(x)]] \\ &= \mathop{\mathbb{E}}_{x_{i_j} \leftarrow U_{b_{i_j}}} [\prod_{j=1}^{r} \hat{\mathbf{B}}^{(2^{i_j})}[\mathrm{INW}_{i_j}(x_{i_j})]] \\ &= \hat{\mathbf{B}}^{(\sum_{j=1}^{r} 2^{i_j})}[\prod_{j=1}^{r} \mathrm{INW}_{i_j}] \end{aligned}$$

where the final line follows from Proposition 5.4, and we conclude by Proposition 5.4. □

We now show that the distribution matrices successively approximate each other with respect to unit circle approximation.

Lemma 5.8. *For all $i \in [q]$,*

$$\mathbf{W}_i \mathop{\approx}^{\circ}_{\frac{\delta}{40 \log(n)}} \mathbf{W}_{i-2}^2.$$

Proof. Let $\hat{\mathbf{A}}_x$ be the transition graph of the length n permutation branching program A_x, which has transition functions $B_1, \dots, B_n, A_0^x$ where $A_0^x(v, b) = (v - 1 + x \mod w) + 1$. We first claim $\hat{\mathbf{B}}[s] = \mathbb{E}_{x=1}^{w} \hat{\mathbf{A}}_x[s]$. Fixing s, clearly $(\hat{\mathbf{A}}_x)_{i,j}[s] = \hat{\mathbf{B}}[s]_{i,j}$ for all x for all blocks $(i, j) \neq (n, 0)$. For the $(n, 0)$ block, for all $x \in [w]$ and $s \in \{0, 1\}$, $\mathbf{A}_x[s]$ is a matching from $v \in [w]$ to $(v - 1 + x \mod w) + 1$. Therefore for any s, the probability over x of reaching state v' from state v is $1/w$ for all $v, v' \in [w]$, so we verify the claim.

Then by Theorem 3.13, for all $x \in [w]$ and $i \in [q]$ we have

$$\mathbb{E}[\hat{\mathbf{A}}_x^{(2^i)}[\mathrm{INW}_i(U_{b_i})]] \mathop{\approx}^{\circ}_{\delta/40 \log(n)} \mathbb{E}[\hat{\mathbf{A}}_x^{(2^i)}[(\mathrm{INW}_{i-1}(U_{b_{i-1}}), \mathrm{INW}_{i-1}(U_{b_{i-1}}))]]$$

Then applying Proposition 3.10 we obtain

$$\begin{aligned} \mathbf{W}_i &= \hat{\mathbf{B}}^{(2^i)}[\mathrm{INW}_i] \\ &= \mathop{\mathbb{E}}_{x=1}^{w} [\mathbb{E}[\hat{\mathbf{A}}_x^{(2^i)}[\mathrm{INW}_i(U_{b_i})]]] \\ &\mathop{\approx}^{\circ}_{\delta/40 \log(n)} \mathop{\mathbb{E}}_{x=1}^{w} [\mathbb{E}[\hat{\mathbf{A}}_j^{(2^i)}[(\mathrm{INW}_{i-1}(U_{b_{i-1}}), \mathrm{INW}_{i-1}(U_{b_{i-1}}))]]] \\ &= \hat{\mathbf{B}}^{(2^i)}[\mathrm{INW}_{i-1}, \mathrm{INW}_{i-1}] \\ &= \mathbf{W}_{i-1}^2 \end{aligned}$$

□

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 [AKM+20]. 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.

Definition 5.9. Let $\mathbf{C}_0 = [1]$ and given $\mathbf{C}_i$ let $\mathbf{C}_{i+1} = \begin{bmatrix} 0 & \mathbf{C}_i \\ \mathbf{I}_{2^i} & 0 \end{bmatrix}$.

We now define the weighted Laplacian of the cycle-lift of length n of $\mathbf{W}_0$.

Definition 5.10. Given $n \in \mathbb{N}$, $\delta \in (0, 1/2)$ and a length n permutation BP $\mathbf{B}$, define $c = 1 - \frac{1}{n+1}$. Let $\hat{\mathbf{B}}$ and $\mathbf{W}_0$ be defined as in Definition 5.6 with $\delta = \delta$. Define the Laplacian of the cycle-lift 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 [AKM+20] except we multiply $\mathbf{W}_0$ by a small factor. This makes $\mathbf{L}$ invertible, making the analysis cleaner and providing a bound on the condition number of $\mathbf{L}$ that is independent of the width of the branching program. We now detail the decomposition of this laplacian using repeated Schur complements, identical to that of Section 6 of [AKM+20].

Let H be the set of indices $\{2^{q-1}N, 2^{q-1}N + 2, \dots, 2^q N - 1\}$, so that the cycle $\mathbf{C}_q$ alternates between H and H^c. We get the following nice form for the Schur complement of L onto the set H:

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

This is exactly the Laplacian of the cycle lift of $\mathbf{W}_0^2$ 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 $\mathbf{C}_{q-2}$. We repeat this procedure q times to obtain a decomposition of the laplacian in terms of repeated Schur complements. Formally, define $\mathbf{L}^{(0)} = \mathbf{L}$ and for all $i \in [q]$ define

$$\mathbf{L}^{(i)} = \mathbf{X}_1 \dots \mathbf{X}_i \begin{bmatrix} \mathbf{I}_{(2^q - 2^{q-i})N} & 0 \\ 0 & \mathbf{I}_{2^{q-i}} - \mathbf{C}_{q-i} \otimes c^{2^i}\mathbf{W}_i \end{bmatrix} \mathbf{Y}_i \dots \mathbf{Y}_1 \tag{6}$$

Where

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

Given this factorization, it is easy to compute the inverse:

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

Where

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

Note that

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

where the scaling by c_q allows a power series representation, and the dependence on $\mathbf{B}$ falls out.

We now characterize this cycle lift inverse in terms of the base matrices $\mathbf{W}_i$. To do so, we define $\mathbf{D}_r$ for $r \in \{0, \dots, q\}$. Intuitively, $\mathbf{D}_r$ "holds" the layers of the original cycle $\mathbf{C}_q$ that are divisible by 2^r. We start with layer 0 in $\mathbf{D}_q$ and repeatedly lift using $\mathbf{X}_i^{-1}$ and $\mathbf{Y}_i^{-1}$ to construct indices with smaller least significant digits.

Definition 5.11. Let $\mathbf{D}_q = (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} \in \mathbb{R}^{N \times N}$. Given $\mathbf{D}_r \in \mathbb{R}^{N2^{q-r} \times N2^{q-r}}$, define $\mathbf{D}_{r-1} \in \mathbb{R}^{N2^{q+1-r} \times N2^{q+1-r}}$ as

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

With this definition, $\mathbf{D}_0 = (\mathbf{L}^{(q)})^{-1}$.

We index the rows and columns of the $N \times N$ blocks of $\mathbf{D}_r$ by the cycle $\mathbf{C}_{q-r}$, multiplied by 2^r (for instance, $\mathbf{D}_{q-1}$ has block indices $\{0, 2^{q-1}\}$). We now show the inverse has the correct form.

Lemma 5.12. *For all $k \in \{0, \dots, q\}$ and $v, w \in \{0, \dots, 2^{q-k} - 1\} \cdot 2^k$, let $m = w - v \mod 2^q$. Then*

$$(\mathbf{D}_k)_{v,w} = (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^m \mathbf{W}_{i_1} \dots \mathbf{W}_{i_r}$$

where $\sum_{j=1}^{r} 2^{i_j} = m$ and $r \le 2(q-k)$. Furthermore, given v, w the indices $i_1, \dots, i_r$ are computable in space $O(\log(n))$.

Proof. We prove this using backwards induction from $\mathbf{D}_q$. For the base case, there is only one block $v = w = 0$ and $(\mathbf{D}_q)_{0,0} = (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1}$ as claimed. Now assume that the indices fixed by $\mathbf{D}_r$ have the claimed form. Fix any indices $v, w \in \{0, \dots, 2^{q-r}\}$ and consider the 2×2 submatrix of $\mathbf{D}_{r-1}$ induced by the (v, w) entry of each of the four components of the lift. From the definition of $\mathbf{C}_{q-(r-1)}$, this submatrix has indices

$$S = \begin{bmatrix} (v + 2^{r-1}, w + 2^{r-1}) & (v + 2^{r-1}, w) \\ (v, w + 2^{r-1}) & (v, w) \end{bmatrix}.$$

We aim to show the entries of $(\mathbf{D}_{r-1})_{|S}$ have the desired form. From the definition of the indexing we obtain $((\mathbf{C}_{q-r} \otimes \mathbf{I}_N)\mathbf{D}_r)_{v,w} = (\mathbf{D}_r)_{v+2^r,w}$. Recalling the structure of the lift,

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

Note that for $v \neq w$ we have $|v - w| \ge 2^r$ by definition, and $\mathbf{I}_{n+1} \otimes \mathbf{J}_w$ commutes with $\mathbf{W}_i$ for all i by Remark 5.5.

- $(\mathbf{D}_{r-1})_{v,w} = (\mathbf{D}_r)_{v,w}$ has the claimed form by the inductive assumption.
- $(\mathbf{D}_{r-1})_{v,w+2^{r-1}} = (\mathbf{D}_r)_{v,w} c_{r-1}\mathbf{W}_{r-1}$ and so has the claimed form.
- $(\mathbf{D}_{r-1})_{v+2^{r-1},w} = c_{r-1}\mathbf{W}_{r-1}(\mathbf{D}_r)_{v+2^r,w}$ and so has the claimed form.
- For the upper right block where $v \neq w$,

  $$(\mathbf{D}_{r-1})_{v+2^{r-1},w+2^{r-1}} = c_{r-1}\mathbf{W}_{r-1}(\mathbf{D}_r)_{v+2^r,w} c_{r-1}\mathbf{W}_{r-1}$$

  and so has the claimed form.

- Finally, for the upper left block where $v = w$, let $\widetilde{W}$ be the product of $\mathbf{W}_{i_j}$ matrices in $(\mathbf{D}_r)_{v+2^r,v}$, which must have total length $v - (v + 2^r) \mod 2^q = 2^q - 2^r$ by the inductive assumption. Therefore,

$$\begin{aligned}(\mathbf{D}_{r-1})_{v+2^{r-1},w+2^{r-1}} &= \mathbf{I}_N + c_{r-1}\mathbf{W}_{r-1}(\mathbf{D}_r)_{v+2^r,v}c_{r-1}\mathbf{W}_{r-1} \\ &= \mathbf{I}_N + c_{r-1}\mathbf{W}_{r-1}[(\mathbf{I}_N - c_q\mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1}c^{2^q-2^r}\widetilde{W}]c_{r-1}\mathbf{W}_{r-1} \\ &= \mathbf{I}_N + c_q(\mathbf{I}_{n+1} \otimes \mathbf{J}_w)(\mathbf{I}_N - c_q\mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} \\ &= (\mathbf{I}_N - c_q\mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1}\end{aligned}$$

  and so has the claimed form.

It is clear that each lift concatenates at most two matrices $\mathbf{W}_{r-1}$ to any block, so the number of matrices in each product is bounded as desired.

Finally, given $x, y \in \{0, \ldots, 2^{q-(r-1)} - 1\} \cdot 2^{r-1}$ in the index set of $\mathbf{D}_{r-1}$, it is simple to determine which indices x', y' of $\mathbf{D}_r$ were used to lift to $(\mathbf{D}_{r-1})_{x,y}$. Given this, we determine if the lift to $\mathbf{D}_{r-1}$ left and/or right concatenated $\mathbf{W}_{r-1}$ (with corresponding index $r-1$) and recurse. Since this requires storing x, y 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 $\mathbf{B}$, only n and the (fixed) lift procedure. □

Corollary 5.13. *By taking $\mathbf{W}_i = \mathbf{W}_0^{2^i}$ for all i (corresponding to simply writing $\mathbf{L}$ as q successive Schur complements and not replacing higher powers with approximations) we obtain from Lemma 5.12 that*

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

5.2 Applying Richardson Iteration

From the previous subsection, we obtain a matrix $(\mathbf{L}^{(q)})^{-1}$ which we wish to show is a good (enough) preconditioner of $\mathbf{L}$. To do so, we use tools from [CKP+17, AKM+20]. This will allow us to boost the quality of our approximation using Richardson iteration.

Proposition 5.14. *Given $n \in \mathbb{N}$ and $\delta, \varepsilon \in (0, 1/2)$, define $\ell = \lceil \log_{1/\delta}(1/\varepsilon) \rceil$. Fix any length n permutation BP $\mathbf{B}$ and let $\mathbf{L}$ and $\mathbf{L}^{(q)}$ be defined as in Definition 5.10. Then define $\mathbf{Err} = \mathbf{I}_{2^qN} - (\mathbf{L}^{(q)})^{-1}\mathbf{L}$ and*

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

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

To prove this, we recall machinery from [AKM+20]. For all $i \in \{0, \ldots, q\}$, define

$$\mathbf{S}^{(i)} = \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I}_{2^{q-i}N} - \mathbf{C}_{q-i} \otimes c_i\mathbf{W}_i \end{bmatrix} \in \mathbb{R}^{2^qN \times 2^qN} \tag{7}$$

where the 0 padding is added to make the dimensions of the matrices equal. These correspond to the approximate Schur complement blocks used in the cycle decomposition.

Lemma 5.15 ([AKM+20] Lemma 6.7)**.** *Let* $\mathbf{S}^0, \ldots, \mathbf{S}^{(q)}$ *and* $\mathbf{L}^{(0)}, \ldots, \mathbf{L}^{(q)}$ *be defined as in Equation (7) and Equation (6) respectively. Then for*

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

we have:

- *For each* $0 \leq i \leq q$,
$$\left|\left|\mathbf{F}^{+/2}(\mathbf{L} - \mathbf{L}^{(i)})\mathbf{F}^{+/2}\right|\right|_2 \leq \varepsilon/q,$$
- *The final matrix* $\mathbf{L}^{(q)}$ *satisfies*
$$\mathbf{L}^{(q)^T}\mathbf{F}^{+}\mathbf{L}^{(q)} \succeq \frac{1}{40q^2}\mathbf{F}.$$

The lemma is proved in [AKM+20] for Laplacians of cycle lifts of Eulerian graphs without the scaling factor c, but no component of the proof requires this. We reproduce the proof in Appendix A. We now recall a further lemma required to bound the norm of $\mathbf{Err}$.

Lemma 5.16 ([AKM+20] 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*

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

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

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

Proof of Proposition 5.14. We apply Lemma 5.16 with $\mathbf{L} = \mathbf{L}, \widetilde{\mathbf{L}} = \mathbf{L}^{(q)}$ and $\mathbf{F} = \mathbf{F}$, all of which are invertible and thus trivially satisfy the kernel properties, and satisfy the other properties by Lemma 5.15, which gives

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

We then apply Lemma 2.2 with $\mathbf{A} = \mathbf{L}, \mathbf{E} = (\mathbf{L}^{(q)})^{-1}$, $m = \ell$, and norm $||\cdot|| = ||\cdot||_{\mathbf{F}}$, so we obtain

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

For any $v \in \mathbb{R}^{2^qN}$ with $||v||_2 = 1$ we have

$$||\mathbf{L}v||_2 = ||\mathbf{S}^{(0)}v||_2 = ||(\mathbf{I}_{2^qN} - \mathbf{C}_q \otimes c\mathbf{W}_0)v||_2 \geq 1 - c = 1/(n+1).$$

Furthermore for all $i > 0$, $\mathbf{U}_{\mathbf{S}^{(i)}}$ is positive semidefinite. Therefore $\mathbf{F} = \frac{2}{q}\sum_{i=0}^{q} \mathbf{U}_{\mathbf{S}^{(i)}}$ is a symmetric positive definite matrix with minimum singular value $\frac{2}{q}(1-c) = 1/\operatorname{poly}(n)$. Then,

$$\begin{aligned}
||\mathbf{L}^{-1} - \widetilde{\mathbf{L}^{-1}}||_{\max} &\leq ||(\mathbf{L}^{-1} - \widetilde{\mathbf{L}^{-1}})\mathbf{L}\mathbf{L}^{-1}||_2 \\
&\leq ||\mathbf{I}_{2^qN} - \widetilde{\mathbf{L}^{-1}}\mathbf{L}||_2||\mathbf{L}^{-1}||_2 \\
&\leq \operatorname{poly}(n) \cdot ||\mathbf{I}_{2^qN} - \widetilde{\mathbf{L}^{-1}}\mathbf{L}||_{\mathbf{F}} \\
&\leq \operatorname{poly}(n) \cdot \varepsilon
\end{aligned}$$

where the third line follows from Proposition 3.7. □

5.3 Translation to Pseudodistributions

Now that we have a Richardson polynomial approximating the true inverse, as in the path lift case we wish to interpret the output as a pseudodistribution.

Definition 5.17. Given $n \in \mathbb{N}$ and $\delta \in (0, 1/2)$, for all $x, y \in \{0, \ldots, n\}$ let $m = y - x \mod n+1$ and define

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

as the product of PRGs satisfying

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

for any length n permutation BP $\mathbf{B}$ where $(\mathbf{L}^{(q)})^{-1}$ is as defined in Definition 5.10 with $n = n$ and $\delta = \delta$. We remark that $r \leq 2q$ and $G_{x,y}$ is explicit given x, y by Lemma 5.12.

Lemma 5.18. *Given $n \in \mathbb{N}$ and $\delta \in (0, 1/2)$, for all $i, j \in \{0, \ldots, n\}$, let $m = j - i \mod n+1$. Let $G_{x,y}$ be defined as in Definition 5.17 with $n = n$ and $\delta = \delta$ and recall R is the trivial PRG on one bit. Then for any length n permutation BP $\mathbf{B}$, let $\mathbf{Err}$ be defined as in Proposition 5.14. Then,*

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

Proof. The approach is nearly identical to that of the path lift. We detail the case where $i < j$:

$$\begin{aligned} \mathbf{Err}_{i,j} &= -\sum_{k=0}^{n} (\mathbf{L}^{(q)})^{-1}_{i,k} \mathbf{L}_{k,j} \\ &= -[(\mathbf{L}^{(q)})^{-1}_{i,j} \mathbf{L}_{j,j} + (\mathbf{L}^{(q)})^{-1}_{i,j-1} \mathbf{L}_{j-1,j}] \\ &= (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} (-c^{j-i} \hat{\mathbf{B}}^{(m)}[G_{i,j}] + c^{j-i-1} \hat{\mathbf{B}}^{(m-1)}[G_{i,j-1}] c\mathbf{W}_0) \\ &= (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^{j-i} (\hat{\mathbf{B}}^{(m)}[G_{i,j-1}R] - \hat{\mathbf{B}}^{(m)}[G_{i,j}]) \\ &= c^{j-i} \hat{\mathbf{B}}^{(m)}[G_{i,j-1}R - G_{i,j}]. \end{aligned}$$

Where the second line follows from the definition of $\mathbf{L}$, the third from Definition 5.17, and the fifth from Remark 5.5. □

We now index the nonzero summands of the Richardson polynomial.

Lemma 5.19. *Given $n, \ell \in \mathbb{N}$ and $\delta \in (0, 1/2)$, let $G_{x,y}$ be as defined in Definition 5.17 and $\mathbb{V}_{n,\ell}$ as in Definition 5.1. Then for all $\sigma \in \mathbb{V}_{n,\ell}$, define*

$$M_\sigma = (G_{0,\sigma_1 - 1}R - G_{0,\sigma_1}) \prod_{i=1}^{r-1} (G_{\sigma_i, \sigma_{i+1}-1}R - G_{\sigma_i,\sigma_{i+1}}) G_{\sigma_r, n}.$$

Then for any length n permutation BP $\mathbf{B}$, let $\mathbf{Err}$ and $(\mathbf{L}^{(q)})^{-1}$ be defined as in Proposition 5.14. Then,

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

Proof. Fix any $r \in \{0, \dots, \ell\}$, then

$$\begin{aligned}
(\mathbf{Err}^r(\mathbf{L}^{(q)})^{-1})_{0,n} &= \sum_{(i_j)\in\{0..n\}^r} \mathbf{Err}_{0,i_1} \prod_{j=1}^{r-1} \mathbf{Err}_{i_j,i_{j+1}} (\mathbf{L}^{(q)})^{-1}_{i_r,n} \\
&= \sum_{\sigma\in\mathbb{V}_{n,\ell}:|\sigma|=r} \mathbf{Err}_{0,\sigma_1} (\prod_{i=1}^{r-1} \mathbf{Err}_{\sigma_d,\sigma_{d+1}}) (\mathbf{L}^{(q)})^{-1}_{\sigma_r,n} \\
&= \sum_{\sigma\in\mathbb{V}_{n,\ell}:|\sigma|=r} \\
&\quad c^{\sigma_1} \hat{\mathbf{B}}^{(\sigma_1)}[G_{0,\sigma_1-1}R - G_{0,\sigma_1}] \prod_{i=1}^{r-1} c^{\sigma_{i+1}-\sigma_i} \hat{\mathbf{B}}^{(\sigma_{i+1}-\sigma_i)}[G_{\sigma_i,\sigma_{i+1}-1}R - G_{\sigma_i,\sigma_{i+1}}] (\mathbf{L}^{(q)})^{-1}_{\sigma_r,n} \\
&= (\mathbf{I}_N - c_q \mathbf{I}_{n+1} \otimes \mathbf{J}_w)^{-1} c^n \sum_{\sigma\in\mathbb{V}_{n,\ell}:|\sigma|=r} \hat{\mathbf{B}}^{(n)}[M_\sigma].
\end{aligned}$$

Where the second line follows from Lemma 5.18 and Remark 5.5 as $\hat{\mathbf{B}}^{(m)}[s] - \hat{\mathbf{B}}^{(m)}[s'] = 0$ for all $m > n$ and all s, s', so all products of total length greater than n are identically zero and fall out. Then we conclude by taking a sum over r. □

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

Lemma 5.20. *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 5.19. Then for any length n permutation BP $\mathbf{B}$, let $\hat{\mathbf{B}}$ be defined as in Definition 5.3. Then*

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

Proof. Recall $\widetilde{\mathbf{L}^{-1}}, \mathbf{L}$ as defined in Proposition 5.14. We obtain

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

where the first line follows from Proposition 5.14, the second from Lemma 5.19 and Corollary 5.13, the third from Remark 5.5 and Proposition 5.7, and the fourth from our choice of c. □

We are now prepared to prove Theorem 5.2.

Theorem 5.2. *For all $n \in \mathbb{N}$ and $\varepsilon, \delta \in (0, 1/2)$, set $\ell = \lceil \log_{1/\delta}(1/\varepsilon) \rceil$. Then there exists a family $\{M_\sigma : \sigma \in \mathbb{V}_{n,\ell}\}$ of PRPGs, where for all σ, M_σ is a sum over 2^ℓ products of $O(\ell \log(n))$ explicit PRGs, each with seed length $O(\log(n) \log(\log(n)/\delta))$, multiplied by a sign. Furthermore, for any permutation branching program $\mathbf{B}$ of length n and arbitrary width,*

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

Proof. Let $\{M_\sigma : \sigma \in \mathbb{V}_{n,\ell}\}$ be defined as in Lemma 5.19. For all σ, we have that M_σ is a sum over 2^ℓ products of $2\ell \log(n+1)$ explicit PRGs, each of which has seed length $O(\log(n) \log(\log(n)/\delta))$, multiplied by a sign. Finally,

$$\begin{aligned}
\varepsilon \cdot \mathrm{poly}(n) &\geq \left|\left| \sum_{\sigma \in \mathbb{V}_{n,\ell}} \hat{\mathbf{B}}^{(n)}[M_\sigma] - \hat{\mathbf{B}}^{(n)}[U_n] \right|\right|_{\max} \\
&\geq \left|\left| \sum_{\sigma \in \mathbb{V}_{n,\ell}} (\hat{\mathbf{B}}^{(n)}[M_\sigma])_{0,n} - (\hat{\mathbf{B}}^{(n)}[U_n])_{0,n} \right|\right|_{\max} \\
&= \left|\left| \sum_{\sigma \in \mathbb{V}_{n,\ell}} \overline{\mathbf{B}}[M_\sigma] - \overline{\mathbf{B}}[U_n] \right|\right|_{\max}
\end{aligned}$$

Where the first line follows from Lemma 5.20 and the third from Proposition 5.4. □

We now have a set of explicit PRPGs $\{M_\sigma : \sigma \in \mathbb{V}_{n,\ell}\}$ whose sum provides a high quality approximation of an arbitrary permutation branching program of length n. As before, we wish to decrease the seed length of each M_σ.

5.4 Permutation Branching Program Inner Derandomization

Given the approximation bound on the sum over PRPGs M_σ, we wish to decrease the seed length of terms in this sum. Applying Lemma 4.9 would give a nearly-logarithmic dependence on width. This is acceptable for polynomial width permutation BPs, but can avoided entirely with the main result of [HPV21].

We now state the inner derandomization lemma.

Lemma 5.21. *Fix $\gamma \in (0, 1/2)$ and a length n PRPG $M = \sum_{i=1}^{t} \tau_i G_{i_1} \dots G_{i_k}$ where $\tau_i \in \{-1, 1\}$ and for all i_j, G_{i_j} is an explicit PRG with seed length s, and given i, τ_i and the index set i_j are computable in space $O(s)$. Then there exists a t bounded explicit PRPG F with seed length $O(s + \log(k) \log(t \log(k)/\gamma))$ such that for any length n permutation branching program $\mathbf{B}$,*

$$\left|\left|\overline{\mathbf{B}}[M] - \overline{\mathbf{B}}[F]\right|\right|_{\max} \leq \gamma.$$

Proof. Fix an arbitrary permutation branching program $\mathbf{B}$ of length n and width w (and degree 2). By Theorem 1.4 of [HPV21], for all $k, s \in \mathbb{N}$ and $\delta \in (0, 1/2)$ there is $H : \{0,1\}^{s_{\mathrm{INW}}} \to (\{0,1\}^s)^k$ such that H is an explicit δ-PRG for permutation branching programs of length k and *degree* 2^s, and H has seed length $s_{\mathrm{INW}} = O(s + \log(k) \log(\log(k)/\delta))$. Let H be the PRG obtained from applying the above theorem with $\delta = \gamma/t$.

Now fix $i \in [t]$ and consider the product $G_{i_1} \dots G_{i_k}$. Let $\{l_{j-1}+1, l_{j-1}+2, \dots, l_j\}$ be the bits of the product output by G_{i_j}, where $l_0 = 0$. Then for all $j \in [k]$, define $\mathbf{B}'_{j-1..j}[s] = \mathbf{B}_{l_{j-1}..l_j}[G_{i_j}(s)]$. 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,

$$\begin{aligned}
\gamma/t &\geq \left|\left|\overline{\mathbf{B}}'[U_{[2^s]^k}] - \overline{\mathbf{B}}'[H]\right|\right|_{\max} \\
&= \left|\left|\prod_{j=1}^{k} \mathbb{E}[\mathbf{B}_{l_{j-1}..l_j}[G_{i_j}(U_s)]] - \mathop{\mathbb{E}}_{x \leftarrow U_{s_{\mathrm{INW}}}} \prod_{j=1}^{k} \mathbf{B}_{l_{j-1}..l_j}[G_{i_j}(H(x)_j)]\right|\right|_{\max} \\
&= \left|\left|\overline{\mathbf{B}}[\prod_{j=1}^{k} G_{i_j}] - \overline{\mathbf{B}}[(\prod_{j=1}^{k} G_{i_j}) \circ H]\right|\right|_{\max}
\end{aligned}$$

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

$$F_i = (\prod_{j=1}^{k} G_{i_j}) \circ H$$

which is explicit by composition of space bounded algorithms and has seed length $s_{\mathrm{INW}} = O(s + \log(k)\log(t\log(k)/\gamma))$. Finally, define $F = \sum_{i=1}^{t} \tau_i F_i$, which is explicit by Definition 3.3 and has seed length $s_{\mathrm{INW}} + O(\log(t)) = O(s + \log(k)\log(t\log(k)/\gamma))$ as desired. Finally,

$$\begin{aligned}
\left|\left|\overline{\mathbf{B}}[M] - \overline{\mathbf{B}}[F]\right|\right| &\leq \sum_{i=1}^{t} \left|\left|\overline{\mathbf{B}}[\tau_i G_{i_1} \dots G_{i_k}] - \overline{\mathbf{B}}[\tau_i F_i]\right|\right| \\
&\leq \frac{\gamma}{t} t
\end{aligned}$$

□

5.5 Proof of Theorem 1.4

We are now prepared to prove the main theorem.

Theorem 5.22 (Restatement of Theorem 1.4)**.** *For all n and $\varepsilon \in (0, 1/2)$, there is an explicit pseudorandom pseudodistribution generator* GEN *with seed length*

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

such that for every permutation branching program $\mathbf{B}$ of length n and arbitrary width,

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

Proof. Let $\mathbf{B}$ be any permutation branching program of length n. Applying Theorem 5.2 with $\varepsilon = \varepsilon/\operatorname{poly}(n)$ 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 family $\{M_\sigma : \sigma \in \mathbb{V}_{n,\ell}\}$ jointly satisfying

$$\left|\left|\sum_{\sigma \in \mathbb{V}_{n,\ell}} \overline{\mathbf{B}}[M_\sigma] - \overline{\mathbf{B}}[U_n]\right|\right|_{\max} \leq \varepsilon/2.$$

For all $\sigma \in \mathbb{V}_{n,\ell}$, let GEN_σ be the explicit PRPG obtained from applying Lemma 5.21 with $M = M_\sigma$, $t = 2^\ell$, $k = 2\log(n+1)\ell$, and error parameter $\gamma = \varepsilon/2|\mathbb{V}_{n,\ell}|$.

Finally, define

$$\mathrm{GEN} = \sum_{\sigma \in \mathbb{V}_{n,\ell}} \mathrm{GEN}_\sigma,$$

then applying the triangle inequality and unwinding definitions, we obtain,

$$\begin{aligned}
\left|\left|\overline{\mathbf{B}}[\mathrm{GEN}] - \overline{\mathbf{B}}[U_n]\right|\right|_{\max} &= \left|\left| \sum_{\sigma \in \mathbb{V}_{n,\ell}} \overline{\mathbf{B}}[\mathrm{GEN}_\sigma] - \overline{\mathbf{B}}[U_n] \right|\right|_{\max} \\
&\leq \sum_{\sigma \in \mathbb{V}_{n,\ell}} \left|\left|\overline{\mathbf{B}}[\mathrm{GEN}_\sigma] - \overline{\mathbf{B}}[M_\sigma]\right|\right|_{\max} + || \sum_{\sigma \in \mathbb{V}_{n,\ell}} \overline{\mathbf{B}}[M_\sigma] - \overline{\mathbf{B}}[U_n]||_{\max} \\
&\leq \frac{\varepsilon}{2|\mathbb{V}_{n,\ell}|}|\mathbb{V}_{n,\ell}| + \varepsilon/2 \\
&= \varepsilon.
\end{aligned}$$

Where the third line follows from Lemma 5.21 and Theorem 5.2.

It remains to show seed length and explicitness. By Lemma 5.21, we have that for all σ, GEN_σ is explicit and has seed length

$$s_{\mathrm{term}} = O(\log(n)\log(\log(n)/\delta) + \log(\log(n)\ell)(\ell + \log(|\mathbb{V}_{n,\ell}|/\varepsilon))).$$

Thus GEN has seed length $s = s_{\mathrm{term}} + s_{\mathrm{sum}} = s_{\mathrm{term}} + O(\log(|\mathbb{V}_{n,\ell}|))$ and is explicit by Definition 3.3.

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

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

which implies $\log|\mathbb{V}_{n,\ell}| = O(\ell\log(n)) = O(\log(n)\sqrt{\log(n/\varepsilon)}/\sqrt{\log\log(n/\varepsilon)})$, which implies

$$\begin{aligned}
s &= s_{\mathrm{sum}} + s_{\mathrm{term}} \\
&= O(\log(n)\log\log(n) + \log(n)\sqrt{\log(n/\varepsilon)\log\log(n/\varepsilon)} + \log\log(n/\varepsilon)(\ell + \log(|\mathbb{V}_{n,\ell}|/\varepsilon))) \\
&= O(\log(n)\sqrt{\log(n/\varepsilon)\log\log(n/\varepsilon)} + \frac{\log\log(n/\varepsilon)}{\sqrt{\log\log(n/\varepsilon)}}\sqrt{\log(n/\varepsilon)}\log(n) + \log\log(n/\varepsilon)\log(1/\varepsilon)) \\
&= O(\log(n)\sqrt{\log(n/\varepsilon)}\sqrt{\log\log(n/\varepsilon)} + \log(1/\varepsilon)\log\log(n/\varepsilon)).
\end{aligned}$$

□

6 Acknowledgements

We thank Jack Murtagh and Sumegha Garg for insightful discussions.

References

[Agr19] Rohit Agrawal. Samplers and extractors for unbounded functions. In Dimitris Achlioptas and László A. Végh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019, September 20-22, 2019, Massachusetts Institute of Technology, Cambridge, MA, USA*, volume 145 of *LIPIcs*, pages 59:1–59:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[AKM+20] 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 *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1295–1306. IEEE, 2020.

[BCG18] Mark Braverman, Gil Cohen, and Sumegha Garg. Hitting sets with near-optimal error for read-once branching programs. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 353–362. ACM, 2018.

[BDVY09] Andrej Bogdanov, Zeev Dvir, Elad Verbin, and Amir Yehudayoff. Pseudorandomness for width 2 branching programs. *Electronic Colloquium on Computational Complexity (ECCC)*, 16:70, 2009.

[Bla18] Jaroslaw Blasiok. Optimal streaming and tracking distinct elements with high probability. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2432–2448. SIAM, 2018.

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.

[BRRY10] Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. Pseudorandom generators for regular branching programs. In *FOCS*, pages 40–47. IEEE Computer Society, 2010.

[CKP+17] Michael B Cohen, Jonathan Kelner, John Peebles, Richard Peng, Anup B Rao, Aaron Sidford, and Adrian Vladu. Almost-linear-time algorithms for markov chains and new spectral primitives for directed graphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 410–419. ACM, 2017.

[CL20] Eshan Chattopadhyay and Jyun-Jie Liao. Optimal error pseudodistributions for read-once branching programs. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 25:1–25:27. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

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

[GMR+12] Parikshit Gopalan, Raghu Meka, Omer Reingold, Luca Trevisan, and Salil Vadhan. Better pseudorandom generators via milder pseudorandom restrictions. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS '12)*. IEEE, 20–23 October 2012.

[Gol97] Oded Goldreich. A sample of samplers - a computational perspective on sampling (survey). *Electronic Colloquium on Computational Complexity (ECCC)*, 4(20), 1997.

[HPV21] William M. Hoza, Edward Pyne, and Salil P. Vadhan. Pseudorandom generators for unbounded-width permutation branching programs. In James R. Lee, editor, *12th*

Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference, volume 185 of *LIPIcs*, pages 7:1–7:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[INW94] Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 356–364, Montréal, Québec, Canada, 23–25 May 1994.

[KNP11] Michal Koucký, Prajakta Nimbhorkar, and Pavel Pudlák. Pseudorandom generators for group products: extended abstract. In Lance Fortnow and Salil P. Vadhan, editors, *STOC*, pages 263–272. ACM, 2011.

[MRT19] Raghu Meka, Omer Reingold, and Avishay Tal. Pseudorandom generators for width-3 branching programs. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 626–637. ACM, 2019.

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

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, October 1994.

[RSV13] Omer Reingold, Thomas Steinke, and Salil Vadhan. Pseudorandomness for regular branching programs via Fourier analysis. In Sofya Raskhodnikova and José Rolim, editors, *Proceedings of the 17th International Workshop on Randomization and Computation (RANDOM '13)*, volume 8096 of *Lecture Notes in Computer Science*, pages 655–670. Springer-Verlag, 21–23 August 2013. Full version posted as ECCC TR13-086 and arXiv:1306.3004 [cs.CC].

[RTV06] Omer Reingold, Luca Trevisan, and Salil Vadhan. Pseudorandom walks in regular digraphs and the RL vs. L problem. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC '06)*, pages 457–466, 21–23 May 2006. Preliminary version as *ECCC* TR05-22, February 2005.

[RV05] Eyal Rozenman and Salil Vadhan. Derandomized squaring of graphs. In *Proceedings of the 8th International Workshop on Randomization and Computation (RANDOM '05)*, number 3624 in Lecture Notes in Computer Science, pages 436–447, Berkeley, CA, August 2005. Springer.

[Ste12] Thomas Steinke. Pseudorandomness for permutation branching programs without the group theory. Technical Report TR12-083, Electronic Colloquium on Computational Complexity (ECCC), July 2012.

[SZ99] Michael Saks and Shiyu Zhou. $\mathrm{BP_HSPACE}(S) \subseteq \mathrm{DSPACE}(S^{3/2})$. *Journal of Computer and System Sciences*, 58(2):376–403, 1999.

[SZ11] Jirí Síma and Stanislav Zák. Almost k-wise independent sets establish hitting sets for width-3 1-branching programs. In Alexander S. Kulikov and Nikolay K. Vereshchagin, editors, *CSR*, volume 6651 of *Lecture Notes in Computer Science*, pages 120–133. Springer, 2011.

[Vad12] Salil P Vadhan. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012.

[Yao82] Andrew C. Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, Chicago, Illinois, 3–5 November 1982. IEEE.

[Zuc97] David Zuckerman. Randomness-optimal oblivious sampling. *Random Structures & Algorithms*, 11(4):345–367, 1997.

A Proof of Lemma 5.15

Here we reproduce the proof of [AKM+20] Lemma 6.7 to verify our claim.

Lemma A.1 (CKKPPRS18 Lemma 2.3)**.** *Consider a sequence of m-by-m matrices $\mathbf{S}^{(0)}, \ldots, \mathbf{S}^{(m)}$ 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 $\mathbf{S}^{(0)}$.*
3. *The symmetrization of each $\mathbf{S}^{(i)}$, denoted $\mathbf{U}_{\mathbf{S}^{(i)}}$, is positive semi-definite.*

Let $\mathbf{M} = \mathbf{M}^{(0)} = \mathbf{S}^{(0)}$ and define $\mathbf{M}^{(1)}, \ldots, \mathbf{M}^{(m)}$ 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 $1 = i_0 < i_1 < \ldots < i_{p_{\max}}$ associated scaling parameters $0 < \theta_0, \ldots, \theta_{p_{\max}-1} < 1/2$ such that $\sum_{p=0}^{p_{\max}-1} \theta_p = 1$, and some global error $0 < \varepsilon < 1/2$, we have for every $0 \le p < p_{\max}$:

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

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:

1. *For each $0 \le i \le p_{\max}$,*
$$||\mathbf{F}^{+/2}(\mathbf{M} - \mathbf{M}^{(i)})\mathbf{F}^{+/2}||_2 \le \varepsilon$$
2. *The final matrix $\mathbf{M}^{(p_{\max})}$ satisfies*
$$\mathbf{M}^{(p_{\max})^T}\mathbf{F}^{+}\mathbf{M}^{(p_{\max})} \succeq \frac{1}{40 p_{\max}^2}\mathbf{F}.$$

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

Lemma A.2 ([AKM+20] Lemma D.4)**.** *Let $\mathbf{L}^{(i)}$ be the $2^q N \times 2^q N$ matrices defined in Section 5, then*

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

Lemma A.3 ([AKM+20] Corollary 4.6)**.** *Let $\widetilde{\mathbf{W}}, \mathbf{W} \in \mathbb{C}^{N\times N}$ be possibly asymmetric matrices such that $\widetilde{\mathbf{W}} \overset{\circ}{\approx}_\varepsilon \mathbf{W}$. For all $k \in \mathbb{N}$ let $C_{(k)}$ be the transition matrix for the directed cycle on k vertices. Then $C_{(k)} \otimes \widetilde{\mathbf{W}} \overset{\circ}{\approx}_\varepsilon C_{(k)} \otimes \mathbf{W}$.*

Lemma A.4 ([AKM+20] Lemma 3.8)**.** *Let $\widetilde{\mathbf{W}}, \mathbf{W} \in \mathbb{C}^{N\times N}$ be possibly asymmetric matrices. Then $\widetilde{\mathbf{W}} \overset{\circ}{\approx}_\varepsilon \mathbf{W}$ if and only if $\left|\left|\mathbf{U}^{+/2}_{\mathbf{I}_N-\mathbf{W}}(\widetilde{\mathbf{W}}-\mathbf{W})\mathbf{U}^{+/2}_{\mathbf{I}_N-\mathbf{W}}\right|\right|_2 \leq \varepsilon$.*

We are now prepared to prove the result.

Lemma 5.15 ([AKM+20] Lemma 6.7)**.** *Let $\mathbf{S}^0, \ldots, \mathbf{S}^{(q)}$ and $\mathbf{L}^{(0)}, \ldots, \mathbf{L}^{(q)}$ be defined as in Equation (7) and Equation (6) respectively. Then for*

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

we have:

- *For each $0 \leq i \leq q$,*
$$\left|\left|\mathbf{F}^{+/2}(\mathbf{L}-\mathbf{L}^{(i)})\mathbf{F}^{+/2}\right|\right|_2 \leq \varepsilon/q,$$
- *The final matrix $\mathbf{L}^{(q)}$ satisfies*
$$\mathbf{L}^{(q)^T}\mathbf{F}^{+}\mathbf{L}^{(q)} \succeq \frac{1}{40q^2}\mathbf{F}.$$

Proof. From $\mathbf{S}^{(i)}$'s and $\mathbf{L}^{(i)}$'s we build a sequence of $\hat{\mathbf{S}}^{(j)}$'s and $\hat{\mathbf{M}}^{(j)}$'s that satisfy the conditions of Lemma A.1, 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

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

and

$$\hat{\mathbf{M}}^{(h+1)} = \hat{\mathbf{M}}^{(h)} + (\hat{\mathbf{S}}^{(h+1)} - \mathrm{Sc}(\hat{\mathbf{M}}^{(h)}, [h+1, 2^qN]))$$

Note that $\hat{\mathbf{S}}$'s satisfy all the three premises in Lemma A.1. First $\hat{\mathbf{S}}^{(i)}$ has non-zero entries only on the indices $[i+1, 2^qN]$. Further as all $\hat{\mathbf{S}}^{(i)}$'s are symmetric diagonally dominant due to the scaling factor, the kernel properties 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's $\mathbf{L}^{(i+1)}$ approximates $\mathbf{L}^{(i)}$ in the norm defined by $\mathbf{U}_{\mathbf{S}^{(i)}}$. By Lemma A.2,

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

Now, given $c_{i+1}\mathbf{W}_{i+1} \overset{\circ}{\approx}_{\varepsilon/q} c_{i+1}\mathbf{W}_i^2$, by Lemma A.3 and Lemma A.4 we obtain

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

where H_i are the indices used for the ith Schur complement. Then since $\mathbf{U}_{\mathrm{Sc}(\mathbf{S}^{(i)},H_i)} \preceq 2\mathbf{U}_{\mathbf{S}^{(i)}}$,

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

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

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

Thus by Lemma A.1 for $\mathbf{F} = \frac{2}{q}\sum_{i=0}^{q}\mathbf{U}_{\mathbf{S}^{(i)}}$ we obtain

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

and

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

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il