Electronic Colloquium on Computational Complexity, Report No. 114 (2023)

Weighted Pseudorandom Generators via Inverse Analysis of Random Walks and Shortcutting

Lijie Chen* William M. Hoza† Xin Lyu‡ Avishay Tal§ Hongxun Wu¶

August 8, 2023

Abstract

A *weighted pseudorandom generator* (WPRG) is a generalization of a pseudorandom generator (PRG) in which, roughly speaking, probabilities are replaced with weights that are permitted to be positive or negative. We present new explicit constructions of WPRGs that fool certain classes of standard-order read-once branching programs. In particular, our WPRGs fool *width*-3 programs, constant-width *regular* programs, and *unbounded-width permutation* programs with a single accepting vertex. In all three cases, the seed length is $\widetilde{O}\left(\log n \cdot \sqrt{\log(1/\varepsilon)} + \log(1/\varepsilon)\right)$, where n is the length of the program and ε is the error of the WPRG.

For comparison, for all three of these models, the best explicit unweighted PRGs known have seed length $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$ (Meka, Reingold, and Tal STOC 2019; Braverman, Rao, Raz, and Yehudayoff SICOMP 2014; Hoza, Pyne, and Vadhan ITCS 2021). Our WPRG seed length is superior when ε is small. For the case of unbounded-width permutation programs, Pyne and Vadhan previously constructed a WPRG with a seed length that is similar to ours (CCC 2021), but their seed length has an extra additive $\log^{3/2} n$ term, so our WPRG is superior when $\varepsilon \gg 1/n$.

Our results are based on a new, general framework for error reduction. Our framework builds on the remarkable recent work by Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan (FOCS 2020) that gave a near-logarithmic space algorithm for estimating random walk probabilities in Eulerian digraphs with high precision. Our framework centers around the "inverse analysis" of random walks and a key combinatorial structure termed "shortcut graphs." Using our new framework and the recent notion of singular value approximation (Ahmadinejad, Peebles, Pyne, Sidford, and Vadhan arXiv 2023), we also present an alternative, simpler proof of Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's main theorem. Compared to the original proof, our new proof avoids much of the sophisticated machinery that was imported from recent work on fast Laplacian solvers.

*Miller Institute for Basic Research in Science, University of California, Berkeley. Email: wjmzmbr@gmail.com. Supported by a Miller Research Fellowship.

†Department of Computer Science, University of Chicago. Email: williamhoza@uchicago.edu. This work was done while the author was visiting the Simons Institute for the Theory of Computing.

‡Department of Computer Science, University of California, Berkeley. Email: lyuxin1999@gmail.com. Supported by Avishay Tal's Sloan Research Fellowship and NSF CAREER Award CCF-2145474, and Jelani Nelson's ONR grant N00014-18-1-2562.

§Department of Computer Science, University of California, Berkeley. Email: atal@berkeley.edu. Supported by a Sloan Research Fellowship and NSF CAREER Award CCF-2145474.

¶Department of Computer Science, University of California, Berkeley. Email: wuhx@berkeley.edu. Supported by Avishay Tal's Sloan Research Fellowship, NSF CAREER Award CCF-2145474, and Jelani Nelson's ONR grant N00014-18-1-2562.

ISSN 1433-8092

Contents

1 Introduction

What is the intrinsic relationship between randomness and space as computational resources? The famous "$\mathsf{L} = \mathsf{BPL}$" conjecture says that for every halting randomized decision algorithm using $S \geq \log n$ bits of space, there is a deterministic algorithm that solves the same problem using $O(S)$ bits of space: randomization buys at most a constant factor in terms of space complexity. The challenge of derandomizing efficient algorithms is well-motivated, because high-quality random bits are not always available easily and without cost.

A traditional approach to derandomization is to try to design sufficiently powerful *pseudorandom generators* (PRGs).

Definition 1.1 (PRGs)**.** *Let $n \in \mathbb{N}$, let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$, and let $\varepsilon > 0$. An* ε-PRG *for $\mathcal{F}$ is a function $\mathcal{G}\colon \{0,1\}^s \to \{0,1\}^n$ such that for every $f \in \mathcal{F}$, we have*

$$|\mathbb{E}[f(\mathcal{G}(U_s))] - \mathbb{E}[f]| \leq \varepsilon.$$

(Above, U_s denotes the uniform distribution over $\{0,1\}^s$, and $\mathbb{E}[f]$ is a shorthand for $\mathbb{E}[f(U_n)]$.) The parameter s is called the seed length *of the PRG. We also say that $\mathcal{G}$* fools *$\mathcal{F}$ with error ε, or ε-fools $\mathcal{F}$.*

For the purpose of derandomizing space-bounded computation, the appropriate class $\mathcal{F}$ consists of polynomial-width standard-order *read-once branching programs* (ROBPs).

Definition 1.2 (Standard-order ROBPs)**.** *Let $w, n \in \mathbb{N}$. A width-w length-n* standard-order ROBP *is a directed graph B. The vertex set consists of $n+1$* layers*, $V = V^{(0)} \cup V^{(1)} \cup \cdots \cup V^{(n)}$, where $|V^{(i)}| \leq w$ for each i. We usually assume without loss of generality that $|V^{(i)}| = w$. Every vertex $v \in V^{(i)}$ with $i < n$ has two outgoing edges, one labeled 0 and the other labeled 1, leading to $V^{(i+1)}$. There is a designated* start vertex *$v_{\text{start}} \in V^{(0)}$, and there is a set of designated* accepting vertices *$V_{\text{accept}} \subseteq V^{(n)}$. The program computes a Boolean function $B\colon \{0,1\}^n \to \{0,1\}$ as follows. Given an input $x \in \{0,1\}^n$, let $v_{\text{start}} = v_0, v_1, v_2, \ldots, v_n$ be the unique path such that for each $i \in [n]$, there is an edge from v_{i-1} to v_i with label x_i. If $v_n \in V_{\text{accept}}$, then we set $B(x) = 1$, and otherwise $B(x) = 0$.*

If A is a randomized space-S algorithm, then for each fixed input σ, the function $B(x) = A(\sigma, x)$ (where x denotes the random bits used by A) can be computed by a width-w length-n standard-order ROBP where $w = n = 2^{O(S)}$. Therefore, given an explicit[1] ε-PRG $\mathcal{G}$ for such programs, we could deterministically estimate the acceptance probability of A to within $\pm\varepsilon$ by computing $2^{-s} \cdot \sum_{u \in \{0,1\}^s} A(\sigma, \mathcal{G}(u))$. In particular, explicit PRGs for width-n length-n ROBPs with seed length $O(\log n)$ would imply $\mathsf{L} = \mathsf{BPL}$.

Using the probabilistic method, one can show the existence of non-explicit ε-PRGs for width-w length-n ROBPs with seed length $O(\log(wn/\varepsilon))$. Furthermore, several unconditional constructions of explicit PRGs for standard-order ROBPs are known. Most famously, Nisan designed a PRG that ε-fools width-w length-n ROBPs with seed length $O(\log(wn/\varepsilon) \cdot \log n)$ [Nis92]. Nisan's PRG has found numerous applications, but its seed length is too large to resolve the L vs. BPL problem.

1.1 Weighted PRGs

Nisan's PRG [Nis92] is more than three decades old. Despite much effort, there is still no known explicit PRG for standard-order ROBPs of polynomial width (or even width 4) with a better

[1] For our purposes, a generator $\mathcal{G}\colon \{0,1\}^s \to \{0,1\}^n$ is *explicit* if it can be computed in space $O(s)$. The algorithm for computing $\mathcal{G}(u)$ is given the seed u along with parameters (n, ε, etc.) specifying $\mathcal{G}$ among a relevant family of generators.

seed length. This motivates the search for *alternative approaches* to proving $\mathsf{L} = \mathsf{BPL}$ that do not necessarily require any breakthroughs on the PRG problem.[2] Braverman, Cohen, and Garg introduced one such approach [BCG20], based on the concept of a *weighted PRG* (WPRG).

Definition 1.3 (WPRGs). *Let $n \in \mathbb{N}$, let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$, and let $\varepsilon > 0$. An ε-*WPRG *for $\mathcal{F}$ is a pair $(\mathcal{G}, \mu)$, where $\mathcal{G}\colon \{0,1\}^s \to \{0,1\}^n$ and $\mu\colon \{0,1\}^s \to \mathbb{R}$, such that for every $f \in \mathcal{F}$, we have*

$$\left|\mathbb{E}_{u\in\{0,1\}^s}[f(\mathcal{G}(u)) \cdot \mu(u)] - \mathbb{E}[f]\right| \leq \varepsilon.$$

The parameter s is called the seed length *of the WPRG. We also say that $(\mathcal{G}, \mu)$* fools *$\mathcal{F}$ with error ε, or ε-fools $\mathcal{F}$.*

Crucially, the weights $\mu(u)$ are allowed to be negative. (Indeed, WPRGs with nonnegative weights are essentially equivalent to unweighted PRGs [PV21, Appendix C].) Intuitively, this means that we are considering the expectation of f with respect to a sparse input "distribution" in which *some probabilities are negative*. For this reason, WPRGs are also known as *pseudorandom pseudodistribution generators* [BCG20].

Because we are allowed to use negative weights, constructing WPRGs is potentially easier than constructing unweighted PRGs. Indeed, Braverman, Cohen, and Garg [BCG20] constructed an explicit WPRG that ε-fools width-w length-n standard-order ROBPs with seed length

$$\widetilde{O}(\log(wn) \cdot \log n + \log(1/\varepsilon)),$$

which is better than Nisan's PRG's seed length when the error parameter ε is very small. A sequence of followup works developed simpler and better WPRG constructions [CL20; CDRST21; PV21; Hoz21], in particular improving the seed length to $O(\log(wn) \cdot \log n + \log(1/\varepsilon))$ [Hoz21]. These examples demonstrate that negative weights open up new avenues for making progress. Furthermore, for a certain class of branching programs, Pyne and Vadhan constructed a WPRG [PV21] with a seed length that is *provably impossible* to achieve via unweighted PRGs [HPV21], demonstrating the *intrinsic* power of negative weights. (Jumping ahead, we improve Pyne and Vadhan's construction in this work. See Section 1.4.)

Despite the presence of negative weights, explicit WPRGs are still useful for derandomization. Indeed, an explicit WPRG for width-n length-n standard-order ROBPs with seed length $O(\log n)$ would imply $\mathsf{L} = \mathsf{BPL}$, just like an explicit unweighted PRG would. The reason is that given such a WPRG, we could deterministically estimate the acceptance probability of a randomized algorithm A by computing $2^{-s} \cdot \sum_{u\in\{0,1\}^s} A(\sigma, \mathcal{G}(u)) \cdot \mu(u)$. Furthermore, WPRGs imply *hitting set generators* (HSGs).

Definition 1.4 (HSGs). *Let $n \in \mathbb{N}$, let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$, and let $\varepsilon > 0$. An ε-*HSG *for $\mathcal{F}$ is a function $\mathcal{G}\colon \{0,1\}^s \to \{0,1\}^n$ such that for every $f \in \mathcal{F}$, if $\mathbb{E}[f] > \varepsilon$, then there exists $u \in \{0,1\}^s$ such that $f(\mathcal{G}(u)) = 1$.*

If $(\mathcal{G}, \mu)$ is an ε-WPRG for $\mathcal{F}$, then $\mathcal{G}$ is an ε-HSG for $\mathcal{F}$ [BCG20]. HSGs have been studied for many decades, perhaps starting with the pioneering work of Ajtai, Komlós, and Szemerédi [AKS87]. It turns out that an explicit HSG for width-n length-n standard-order ROBPs with optimal seed length $O(\log n)$ would already imply $\mathsf{L} = \mathsf{BPL}$ [CH22; PRZ23], just like a PRG or a WPRG. However, in the non-optimal regime (which is the most relevant regime given the present state of knowledge), the known applications of WPRGs exceed the known applications of HSGs. In particular, the current state-of-the-art unconditional derandomization of space-bounded

[2] See Hoza's survey for a discussion of different approaches to proving $\mathsf{L} = \mathsf{BPL}$ [Hoz22].

Seed length	Type	Reference	Notes
$\widetilde{O}(\log(n/\varepsilon))$	HSG	[GMRTV12]	
$\widetilde{O}(\log n \cdot \log(1/\varepsilon))$	PRG	[MRT19]	
$\widetilde{O}\left(\log n \cdot \sqrt{\log(1/\varepsilon)} + \log(1/\varepsilon)\right)$	WPRG	This work	
$O(\log(n/\varepsilon))$	PRG	Folklore	Optimal; non-explicit

Table 1: Pseudorandomness constructions for width-3 standard-order ROBPs. Note that there is also work on width-3 ROBPs in the more challenging arbitrary-order setting [SVW17; MRT19].

computation says that randomized space-S algorithms can be simulated deterministically in space $O(S^{3/2}/\sqrt{\log S})$ [Hoz21]. This result is a recent slight improvement over Saks and Zhou's decades-old $O(S^{3/2})$ bound [SZ99]. The improvement relies on the recent line of work on WPRGs [BCG20; CL20; CDRST21; PV21; Hoz21], and it is not clear how to reproduce the result using HSGs alone.

In summary, it seems that the WPRG concept achieves a "Goldilocks" effect: it is flexible enough to facilitate constructions, yet structured enough to facilitate applications. The WPRG approach to derandomization is therefore highly promising. In this work, we present new and improved constructions of WPRGs for several well-motivated classes of branching programs.

1.2 Width-3 Branching Programs

When $w < n$, width-w length-n standard-order ROBPs do not correspond to uniform algorithms, but they are nevertheless a natural nonuniform model of $(\log w)$-space computation. We emphasize that the transitions may vary from one layer to the next, which can be interpreted as meaning that the program has access to a "clock," in addition to its $(\log w)$-bit workspace.

For width-2 programs, explicit PRGs are known with optimal seed length $O(\log(n/\varepsilon))$ [SZ95; BDVY13; HH23]. The width-3 case is at the frontier of current knowledge. Meka, Reingold, and Tal designed an explicit ε-PRG for width-3 standard-order ROBPs with seed length $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$ [MRT19]. When the error parameter ε is constant, Meka, Reingold, and Tal's seed length is near optimal, but when $\varepsilon = 1/\mathrm{poly}(n)$, their PRG does not beat Nisan's $O(\log^2 n)$ seed length. To address this issue, we present a WPRG that fools width-3 standard-order ROBPs with error $1/\mathrm{poly}(n)$ and seed length $\widetilde{O}(\log^{3/2} n)$. More generally, we achieve the following parameters.

Theorem 1.5 (WPRG for width-3 ROBPs)**.** *For every $n \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit ε-WPRG with seed length*

$$\widetilde{O}\left(\log n \cdot \sqrt{\log(1/\varepsilon)} + \log(1/\varepsilon)\right)$$

that ε-fools width-3 length-n standard-order ROBPs.

Note that near-optimal HSGs for width-3 standard-order ROBPs were already known; see Table 1.

1.3 Regular Branching Programs

Unfortunately, the state of the art for width-4 standard-order ROBPs is the same as that for the polynomial-width case. On the bright side, better results are known for narrow ROBPs that also

satisfy certain structural restrictions such as *regularity*.

Definition 1.6 (Regular ROBPs)**.** *Let B be a standard-order ROBP with vertex set $V = V^{(0)} \cup \cdots \cup V^{(n)}$. We say that B is* regular *if every vertex $v \in V \setminus V^{(0)}$ has precisely two incoming edges.*

Regular ROBPs have been the subject of intense study [RTV06; BRRY14; De11; RSV13; BHPP22; LPV22]. One reason to study regular ROBPs is that there is a reduction from the general case to the regular case. Indeed, Lee, Pyne, and Vadhan recently showed that if a function $f\colon \{0,1\}^n \to \{0,1\}$ can be computed by a standard-order ROBP of width w, then it can also be computed by a standard-order *regular* ROBP of width $O(wn)$ [LPV23]. (This is an improvement over previous reductions [RTV06; BHPP22].) Consequently, optimal explicit WPRGs for polynomial-width standard-order *regular* ROBPs would imply optimal explicit WPRGs for polynomial-width standard-order *non-regular* ROBPs.

Braverman, Rao, Raz, and Yehudayoff designed an explicit ε-PRG for width-w length-n standard-order regular ROBPs with seed length $\widetilde{O}(\log(w/\varepsilon) \cdot \log n)$ [BRRY14]. (See also De's followup work [De11].) When the width parameter w and the error parameter ε are both constant, Braverman, Rao, Raz, and Yehudayoff's seed length is near-optimal. However, when $\varepsilon = 1/\mathrm{poly}(n)$, their seed length is no better than Nisan's $O(\log^2 n)$ seed length, even if we hold w constant. To address this issue, we present a WPRG that fools constant-width standard-order regular ROBPs with error $1/\mathrm{poly}(n)$ and seed length $\widetilde{O}(\log^{3/2} n)$. More generally, our WPRG has the following parameters.

Theorem 1.7 (WPRG for regular ROBPs)**.** *For every $n, w \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit ε-WPRG with seed length*

$$\widetilde{O}\left(\log n \cdot \left(\log w + \sqrt{\log(1/\varepsilon)}\right) + \log(1/\varepsilon)\right)$$

that ε-fools width-w length-n standard-order regular ROBPs.

Our seed length matches that of an HSG for the same class that Bogdanov, Hoza, Prakriya, and Pyne recently constructed [BHPP22] (ignoring $\log\log$ factors). In turn, their construction is the best HSG known for this class (again, ignoring $\log\log$ factors). See Table 2.

1.4 Unbounded-Width Permutation Branching Programs

Definition 1.8 (Permutation ROBPs)**.** *Let B be a standard-order ROBP with vertex set $V = V^{(0)} \cup \cdots \cup V^{(n)}$. We say that B is a* permutation ROBP *if every vertex $v \in V \setminus V^{(0)}$ has precisely two incoming edges, and those two incoming edges have distinct labels.*

In other words, between every two adjacent layers of a permutation ROBP, the edges labeled 0 form a matching, as do the edges labeled 1. Permutation ROBPs are thus a subclass of regular ROBPs. The first sequence of papers studying permutation ROBPs focused on the constant-width regime [BV10; De11; KNP11; Ste12; RSV13; CHHL19]. In recent years, there has been another wave of interest in permutation ROBPs, but this time, the focus is on programs of *unbounded width* with only one accepting vertex [HPV21; PV21; PV22; BHPP22; GV22; LPV22]. This intriguing model cannot be considered a model of "space-bounded" computation anymore; instead, it is a certain type of "reversible" computation. Still, we hope that studying this model can shed light on the L vs. BPL problem. Studying unbounded-width models has already been a fruitful approach for proving new results about standard bounded-width models [PV21; GV22; LPV22]. Indeed, in general, results for unbounded-width models typically imply corresponding results for standard

Seed length	Type	Reference	Notes
$\widetilde{O}(\log(w/\varepsilon) \cdot \log n)$	PRG	[BRRY14]	
$O(w \cdot \log n)$ for all $\varepsilon > 0$	HSG	[BRRY14]	
$\widetilde{O}\left(\log n \cdot \left(\log w + \sqrt{\log(1/\varepsilon)}\right) + \log(1/\varepsilon)\right)$	HSG	[BHPP22]	
$\widetilde{O}\left(\log n \cdot \left(\log w + \sqrt{\log(1/\varepsilon)}\right) + \log(1/\varepsilon)\right)$	WPRG	This work	
$O(\log(wn/\varepsilon))$	PRG	Folklore	Optimal; non-explicit

Table 2: Pseudorandomness constructions for width-w length-n standard-order regular ROBPs. Note that there is also work on the intriguing setting of unbounded-width programs with only one accepting vertex, as well as the challenging arbitrary-order setting [BHPP22; CLTW23]. Furthermore, Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan gave a near-optimal non-black-box algorithm for estimating the acceptance probability of a given standard-order regular ROBP [AKMPSV20]; see Section 1.5.

width-w models, with a factor-of-w loss in the error parameter. Additionally, the unbounded-width model serves as a valuable "test-bed" for studying different notions of pseudorandomness.

Hoza, Pyne, and Vadhan designed an ε-PRG for these programs with seed length $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$ [HPV21], building heavily on prior work by Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan [AKMPSV20]. Importantly, Hoza, Pyne, and Vadhan also proved a near-matching seed length *lower bound*: every ε-PRG for this model has seed length at least $\widetilde{\Omega}(\log n \cdot \log(1/\varepsilon))$ [HPV21]. In contrast, Pyne and Vadhan designed an ε-WPRG for this model with seed length $\widetilde{O}(\log n \cdot \sqrt{\log(n/\varepsilon)} + \log(1/\varepsilon))$ [PV21]. In particular, when $\varepsilon = 1/\mathrm{poly}(n)$, the optimal seed length for unweighted PRGs is $\Theta(\log^2 n)$, whereas Pyne and Vadhan's WPRG seed length is only $\widetilde{O}(\log^{3/2} n)$. As mentioned previously, these results demonstrate that WPRGs have an intrinsic advantage over unweighted PRGs in this case.

A weakness of Pyne and Vadhan's WPRG [PV21] is that the seed length is always at least $\log^{3/2} n$. We present a new WPRG that smoothly interpolates between Hoza, Pyne, and Vadhan's $\widetilde{O}(\log n)$ seed length in the constant-error regime [HPV21] and Pyne and Vadhan's $\widetilde{O}(\log^{3/2} n)$ seed length in the inverse-polynomial-error regime [PV21].

Theorem 1.9 (WPRG for unbounded-width permutation ROBPs)**.** *For every $n \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit WPRG with seed length*

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

that ε-fools unbounded-width standard-order permutation ROBPs with a single accept state.

Thus, our work strengthens the known separation between WPRGs and unweighted PRGs. Note that our seed length is always at least as good as Hoza, Pyne, and Vadhan's PRG seed length [HPV21] (ignoring $\log\log$ factors). See Table 3 for a summary. We also remark that as a simple corollary of Theorem 1.9, we obtain an explicit WPRG for width-w standard-order permutation ROBPs (with an unbounded number of accepting vertices) with seed length

$$\widetilde{O}\left(\log n \cdot \sqrt{\log(w/\varepsilon)} + \log(w/\varepsilon)\right).$$

Seed length	Type	Reference	Notes
$O(\log(n/\varepsilon) \cdot \log n)$	PRG	[De11]	
$\widetilde{O}(\log n \cdot \log(1/\varepsilon))$	PRG	[HPV21]	
$\widetilde{O}\left(\log n \cdot \sqrt{\log(n/\varepsilon)} + \log(1/\varepsilon)\right)$	WPRG	[PV21]	
$\widetilde{O}\left(\log n \cdot \sqrt{\log(1/\varepsilon)} + \log(1/\varepsilon)\right)$	WPRG	This work	
$\widetilde{\Omega}(\log n \cdot \log(1/\varepsilon))$	PRG	[HPV21]	Lower bound
$O(\log(n/\varepsilon))$	HSG	[HPV21]	Optimal; non-explicit
$O(\log(n/\varepsilon))$	Det. sampler[3]	[PV22]	Optimal; non-explicit

Table 3: Pseudorandomness constructions for unbounded-width permutation ROBPs with a single accepting state. Note that there is also work on the more challenging arbitrary-order setting [BHPP22; LPV22; CLTW23]. Note also that the optimal WPRG seed length for this model is unclear.

1.5 A Simpler High-Precision Non-Black-Box Derandomization of Regular ROBPs

In addition to WPRGs, we also study non-black-box derandomization algorithms. In this model, we are given the complete description of an ROBP, and our job is to estimate its acceptance probability. In a remarkable recent work [AKMPSV20], Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan designed an algorithm for estimating the acceptance probability of a given regular ROBP within an inverse polynomial error using $\widetilde{O}(\log(wn))$ bits of space. Formally:

Theorem 1.10 (High-precision non-black-box derandomization of regular ROBPs [AKMPSV20])**.** *There is a deterministic algorithm that uses $\widetilde{O}(\log(wn) \cdot \log\log(1/\varepsilon))$ bits of space and outputs a value that is within $\pm\varepsilon$ of $\mathbb{E}[B]$, given a width-w length-n standard-order regular ROBP B and a value $\varepsilon \in (0,1)$ as inputs.*

Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's work [AKMPSV20] combines techniques from the pseudorandomness literature with sophisticated machinery developed in a sequence of works on faster solvers for Eulerian directed Laplacian systems [CKPPSV16; CKPPRSV17; CKKPPRS18]. Roughly speaking, it introduced three new ideas to the space-bounded derandomization community: (1) the applicability of *Richardson iteration* for error reduction in the space-bounded setting, (2) the notion of the *unit-circle approximation*, and (3) a sophisticated analysis of the INW generator [INW94] via bounding a matrix norm defined by a recursive application of Schur complement (see [AKMPSV20, Theorem 6.1]).

The first two ideas have led to a flurry of exciting new results in space-bounded derandomization [CDRST21; PV21; Hoz21; HPV21; PP22; CDST22], including much of the prior work on WPRGs that we described earlier. In contrast, the third idea has found limited applications so far, despite the fact that it is arguably the core of Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's proof [AKMPSV20]. Only one paper, by Pyne and Vadhan [PV21], has managed to adapt this technique to a new problem.

[3] A *deterministic ε-sampler* for a class $\mathcal{F}$ is a deterministic algorithm SAMP that makes queries to an unknown $f \in \mathcal{F}$ and outputs a real number Samp^f satisfying $|\mathsf{Samp}^f - \mathbb{E}[f]| \leq \varepsilon$. We define the "seed length" of the sampler to be the log of the query complexity. One can show that WPRGs imply deterministic samplers, and deterministic samplers imply HSGs [CH22].

A possible explanation might be that it is not easy to understand the *meaning* of the constructed matrix norm [AKMPSV20, Section 6] in the context of derandomizing ROBPs. Moreover, Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's analysis [AKMPSV20] makes heavy use of complicated linear algebra facts developed in the Laplacian solver literature, and it is unclear how to interpret these facts when applied to the matrix constructed from a branching program.

To remedy this situation, we present a significantly simpler proof of Theorem 1.10. In our proof, we view the recursive Schur complement construction in the original proof as a natural combinatorial "shortcut graph;" see Section 3.1.3 for details. We believe that this proof helps to clarify what exactly the recursive-Schur-complement-matrix-norm really represents in the setting of branching programs. We hope that our proof enables a fuller appreciation of Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's analysis [AKMPSV20] and thereby facilitates the development of new ideas.[4]

Organization. The remainder of this paper is organized as follows. We begin with some definitions and notation in Section 2. Then, in Section 3, we present an informal overview of all of our proofs. Next, in Section 4 and Section 5, we present a general framework that underlies all of our results. In Section 6, we prove Theorem 1.7 (our WPRG for bounded-width regular ROBPs). In Section 7, we prove Theorem 1.5 (our WPRG for width-3 ROBPs). In Section 8, we present our simplified proof of Theorem 1.10 (Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's main result [AKMPSV20]). Finally, in Section 9, we prove Theorem 1.9 (our WPRG for unbounded-width permutation ROBPs).

2 Preliminaries

2.1 Read-Once Branching Programs

Let B be a standard-order ROBP (Definition 1.2) with vertex set $V = V^{(0)} \cup \cdots \cup V^{(n)}$. Let $v_{\text{start}} \in V^{(0)}$ be the start vertex, and let $V_{\text{accept}} \subseteq V^{(n)}$ be the set of accepting vertices.

The transition notation $B[u, x]$. For $\ell \in \{0, \ldots, n\}$, $u \in V^{(\ell)}$, and $x \in \{0,1\}^{\leq n-\ell}$, we let $B[u, x]$ be the vertex $v \in V^{(\ell+|x|)}$ that is reached from u by traversing the edges with labels specified by x. Using this notation, for $x \in \{0,1\}^n$, we have

$$B(x) = 1 \iff B[v_{\text{start}}, x] \in V_{\text{accept}}.$$

The subprogram notation $B^{v \leftarrow u}$. Let $0 \leq \ell \leq r \leq n$, let $u \in V^{(\ell)}$, and let $S \subseteq V^{(r)}$. We define $B^{S \leftarrow u}$ to be the program obtained from B by specifying u as the new start vertex and S as the new set of accepting vertices. We use the following shorthands:

$$B^{v \leftarrow u} := B^{\{v\} \leftarrow u}, \qquad B^{v \leftarrow} := B^{v \leftarrow v_{\text{start}}}, \qquad B^{\leftarrow u} := B^{V_{\text{accept}} \leftarrow u}.$$

For convenience, we think of $B^{S \leftarrow u}$ as a program of length n rather than length $r - \ell$. That is, in $B^{S \leftarrow u}$, the first ℓ transitions are trivial identity layers; the next $r - \ell$ transitions are the same as in

[4] Indeed, all of our new WPRG constructions are inspired by the insights from our simpler proof of Theorem 1.10.

B; and the final $n - r$ transitions are trivial identity layers. Thus, we think of $B^{S \leftarrow u}$ as a Boolean function on n bits, but it only depends on $r - \ell$ of those bits:

$$B^{S \leftarrow u}(x_1, \ldots, x_n) = 1 \iff B[u, (x_{\ell+1}, \ldots, x_r)] \in S.$$

Nevertheless, we will occasionally abuse notation and write $B^{S \leftarrow u}(x_{\ell+1}, \ldots, x_r)$ rather than $B^{S \leftarrow u}(x_1, \ldots, x_n)$.

The matrix notation $\mathbf{B}(x)$. As explained in Definition 1.2, we identify B with a Boolean function $B \colon \{0,1\}^n \to \{0,1\}$. We use boldface $\mathbf{B}$ to denote the matrix-valued function $\mathbf{B} \colon \{0,1\}^n \to \{0,1\}^{V^{(n)} \times V^{(0)}} \cong \{0,1\}^{w \times w}$ given by

$$\mathbf{B}(x)_{v,u} = B^{v \leftarrow u}(x).$$

ROBPs over large alphabets. We occasionally use the standard large-alphabet generalization of Definition 1.2. A width-w length-n standard-order ROBP *over the alphabet* Σ is defined just like Definition 1.2, except that each vertex in $V^{(0)} \cup \cdots \cup V^{(n-1)}$ has $|\Sigma|$ outgoing edges leading to the next layer, labeled with the symbols in Σ. Thus, the program computes a function $B \colon \Sigma^n \to \{0,1\}$. We say that the program is *regular* if each vertex in $V^{(1)} \cup \cdots \cup V^{(n)}$ has precisely $|\Sigma|$ incoming edges, and we say that the program is a *permutation program* if those $|\Sigma|$ incoming edges have distinct labels.

2.2 Linear Algebra

We use boldface to denote matrices. We denote the $w \times w$ identity matrix by $\mathbf{I}_w$. We often simply write $\mathbf{I}$ if the dimension w is clear from context. Recall that every positive semidefinite matrix $\mathbf{M}$ induces a vector "norm" $\|\cdot\|_{\mathbf{M}}$:

Definition 2.1 (Vector seminorm induced by a psd matrix)**.** *Let $\mathbf{M} \in \mathbb{R}^{N \times N}$ be a positive semidefinite matrix. We define a corresponding function $\|\cdot\|_{\mathbf{M}} \colon \mathbb{R}^N \to [0, \infty)$ by the rule*

$$\|x\|_{\mathbf{M}} = \sqrt{x^T \mathbf{M} x}.$$

For example, if $\mathbf{W}$ is a doubly stochastic matrix, then $\mathbf{I} - \mathbf{W}^T\mathbf{W}$ is positive semidefinite, and the corresponding function $\|\cdot\|_{\mathbf{I}-\mathbf{W}^T\mathbf{W}}$ is given by

$$\|x\|^2_{\mathbf{I}-\mathbf{W}^T\mathbf{W}} = \|x\|_2^2 - \|\mathbf{W}x\|_2^2.$$

Observe that we can sometimes have $\|x\|_{\mathbf{M}} = 0$ even if $x \neq 0$. This means that technically, $\|\cdot\|_{\mathbf{M}}$ is a *seminorm* (defined below) rather than a true norm.

Definition 2.2 (Vector seminorm)**.** *A* seminorm *on $\mathbb{R}^N$ is a function $\|\cdot\| \colon \mathbb{R}^N \to [0, \infty)$ satisfying the following properties:*

1. *For every $x, y \in \mathbb{R}^N$, we have $\|x + y\| \leq \|x\| + \|y\|$.*
2. *For every $x \in \mathbb{R}^N$ and every $\lambda \in \mathbb{R}$, we have $\|\lambda x\| = |\lambda| \cdot \|x\|$.*

It is standard that any vector norm $\|\cdot\|$ induces a corresponding matrix norm, namely, the *operator norm*:

$$\|\mathbf{M}\| := \max_{x \neq 0} \frac{\|\mathbf{M}x\|}{\|x\|}.$$

What if we start with a vector *seminorm* $\|\cdot\|$ rather than a true norm? Unfortunately, in this case, the definition above can suffer from divison-by-zero issues, i.e., sometimes we have $\|\mathbf{M}\| = \infty$. For this reason, in this case, the matrix "norm" defined by the equation above is technically not a norm, nor even a seminorm, but rather it is an *extended seminorm*, defined next.

Definition 2.3 (Extended submultiplicative matrix seminorm)**.** *An* extended submultiplicative matrix seminorm *on* $\mathbb{R}^{N\times N}$ *is a function* $\|\cdot\|\colon \mathbb{R}^{N\times N} \to [0,\infty]$ *satisfying the following conditions.*

1. *For every* $\mathbf{A},\mathbf{B} \in \mathbb{R}^{N\times N}$*, if* $\|\mathbf{A}\| < \infty$ *and* $\|\mathbf{B}\| < \infty$*, then* $\|\mathbf{A}+\mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$.
2. *For every* $\mathbf{A},\mathbf{B} \in \mathbb{R}^{N\times N}$*, if* $\|\mathbf{A}\| < \infty$ *and* $\|\mathbf{B}\| < \infty$*, then* $\|\mathbf{A}\cdot\mathbf{B}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{B}\|$.
3. *For every* $\mathbf{M} \in \mathbb{R}^{N\times N}$ *and every nonzero* $\lambda \in \mathbb{R}$*, we have* $\|\lambda \cdot \mathbf{M}\| = |\lambda| \cdot \|\mathbf{M}\|$.
4. *The zero matrix* $\mathbf{0}$ *satisfies* $\|\mathbf{0}\| = 0$.

Definition 2.4 (Extended submultiplicative matrix seminorm induced by a vector seminorm)**.** *Let* $\|\cdot\|\colon \mathbb{R}^N \to [0,\infty)$ *be a vector seminorm. We define an extended submultiplicative matrix seminorm* $\|\cdot\|\colon \mathbb{R}^{N\times N} \to [0,\infty]$ *by the rule*

$$\|\mathbf{M}\| = \min\Big\{\lambda \in \mathbb{R}\cup\{\infty\} : \text{for every } x \in \mathbb{R}^N,\ \|\mathbf{M}\cdot x\| \leq \lambda \cdot \|x\|\Big\}.$$

3 Technical Overview

3.1 Our Error Reduction Framework

Each of our constructions follows the same high-level approach: We start with a construction that has moderate error $\tau \gg \varepsilon$, and then we apply an *error reduction* procedure to decrease the error down to ε. For example, for regular ROBPs, our WPRG construction (Theorem 1.7) works by applying an error reduction procedure to the BRRY PRG [BRRY14]. Many recent works on space-bounded derandomization have developed and applied error reduction procedures [HZ20; AKMPSV20; CDRST21; PV21; Hoz21; BHPP22; PP22; CDST22]. Like most of this prior work, our approach for error reduction is based on the "inverse Laplacian" perspective on space-bounded derandomization, introduced by Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan [AKMPSV20]. We review this perspective next.

3.1.1 The Inverse Laplacian Perspective

Let B be a width-w length-n standard-order ROBP. The vertices of B are denoted as $V(B) := V^{(0)} \cup V^{(1)} \cup \cdots \cup V^{(n)}$ where $V^{(i)} = \{i\cdot w + 1, \ldots, (i+1)\cdot w\}$ for each $i \in \{0,1,\ldots,n\}$. Next, we define $\mathbf{W} \in \mathbb{R}^{(n+1)\cdot w \times (n+1)\cdot w}$ to be the transition matrix of B. Specifically, for every directed edge (u,v) in B, we set $\mathbf{W}_{v,u} = \frac{1}{2}$ (or $\mathbf{W}_{v,u} = 1$ if both outgoing edges of u go to v). [5] Since B only

[5] Note that the index order is (v,u). This convention implies that taking a step in B corresponds to *left*-multiplication by $\mathbf{W}$.

contains edges between pairs of adjacent layers, $\mathbf{W}$ has the following form:

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

In the equation above, $\mathbf{W}_i \in \mathbb{R}^{V^{(i)} \times V^{(i-1)}}$ denotes the transition matrix from the $(i-1)$-th layer of B to the i-th layer. We define the Laplacian of $\mathbf{W}$ as

$$\mathbf{L} \coloneqq \mathbf{I}_{(n+1)w} - \mathbf{W}.$$

Being a unitriangular matrix, $\mathbf{L}$ is invertible. Observe that

$$\mathbf{L}^{-1} = \sum_{i=0}^{n} \mathbf{W}^i = \begin{bmatrix} \mathbf{I}_w & 0 & 0 & \cdots & 0 & 0 \\ \mathbf{W}_{1\leftarrow 0} & \mathbf{I}_w & 0 & \cdots & 0 & 0 \\ \mathbf{W}_{2\leftarrow 0} & \mathbf{W}_{2\leftarrow 1} & \mathbf{I}_w & \cdots & 0 & 0 \\ \vdots & & \ddots & & & \vdots \\ \mathbf{W}_{(n-1)\leftarrow 0} & \mathbf{W}_{(n-1)\leftarrow 1} & \mathbf{W}_{(n-1)\leftarrow 2} & \ddots & \mathbf{I}_w & 0 \\ \mathbf{W}_{n\leftarrow 0} & \mathbf{W}_{n\leftarrow 1} & \mathbf{W}_{n\leftarrow 2} & \cdots & \mathbf{W}_{n\leftarrow (n-1)} & \mathbf{I}_w \end{bmatrix}, \tag{1}$$

where we use $\mathbf{W}_{j\leftarrow i}$ to denote $\mathbf{W}_j \cdot \mathbf{W}_{j-1} \cdots \mathbf{W}_{i+1}$. This matrix, $\mathbf{L}^{-1}$, describes the random walks of all lengths from all starting vertices in B. Looking at (1), we see that $\mathbf{W}_{n\leftarrow 0}$ is a submatrix of $\mathbf{L}^{-1}$, and thus the problem of estimating $\mathbb{E}[B]$ reduces to approximating $\mathbf{L}^{-1}$.

The inverse Laplacian perspective is easiest to understand in the non-black-box setting: if we are given the *description* of the ROBP B, then we can readily compute the Laplacian matrix $\mathbf{L}$, and it makes sense to try to approximately invert it. It turns out that the perspective is also valuable in the black-box setting. To construct WPRGs, our approach will be to first reason in terms of matrix arithmetic, and then "reverse engineer" a WPRG such that all the matrix arithmetic happens in the *analysis* of the WPRG rather than its construction. This technique is due to Cohen, Doron, Renard, Sberlo, and Ta-Shma [CDRST21] and, independently, Pyne and Vadhan [PV21]. For this technical overview, we primarily focus on the matrix arithmetic itself.

3.1.2 Richardson Iteration

With the inverse Laplacian perspective in mind, our high-level plan for proving our results is as follows. First, we will construct a matrix $\widehat{\mathbf{L}}^{-1}$ that approximates $\mathbf{L}^{-1}$ with moderate error.[6] Specifically, we will ensure that $\|\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L}\| \leq \delta$ for a suitable, submultiplicative matrix norm[7] $\|\cdot\|$ and some "moderately small" $\delta < 1$. Then, we will apply a powerful, generic error reduction technique called *Richardson iteration*: for each $m \in \mathbb{N}$, define

$$\mathbf{A}_m = \sum_{i=0}^{m} (\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L})^i \cdot \widehat{\mathbf{L}}^{-1}.$$

[6] We use $\widehat{\mathbf{L}}^{-1}$ to denote our approximate inverse because our matrix $\widehat{\mathbf{L}}^{-1}$ is indeed the *exact* inverse of another matrix $\widehat{\mathbf{L}}$, and $\widehat{\mathbf{L}}$ plays a crucial part in our analysis.

[7] Technically, in some cases we will use an "extended seminorm" rather than a true norm; see Definition 2.3.

It turns out that $\|\mathbf{I} - \mathbf{A}_m\mathbf{L}\| \leq \|\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L}\|^{m+1} \leq \delta^{m+1}$ (see Lemma 4.12). Finally, we will use this low-error approximate inverse $\mathbf{A}_m$ to compute an approximation to $\mathbb{E}[B]$ with low additive error.

Given the plan described above, our main task is to explain how to construct the matrix $\widehat{\mathbf{L}}^{-1}$. The upside of this "Richardson iteration" approach is that the matrix $\widehat{\mathbf{L}}^{-1}$ only needs to be a *moderate-error* approximation. The downside is that $\widehat{\mathbf{L}}^{-1}$ must approximate the *entire* inverse Laplacian matrix $\mathbf{L}^{-1}$, as stipulated by the requirement $\|\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L}\| \leq \delta$. This is true despite the fact that ultimately, we only care about a single $w \times w$ block of $\mathbf{L}^{-1}$. To put it another way, to use Richardson iteration, we must construct a moderate-error collection of estimated acceptance probabilities for *all subprograms* of B, even though we are ultimately only interested in $\mathbb{E}[B]$.

So, how shall we construct this matrix $\widehat{\mathbf{L}}^{-1}$? For each $w \times w$ block $\mathbf{W}_{j\leftarrow i}$ of $\mathbf{L}^{-1}$, we begin by constructing a matrix $\widetilde{\mathbf{W}}_{j\leftarrow i}$ that approximates $\mathbf{W}_{j\leftarrow i}$ with moderate error. For example, in the case of bounded-width standard-order regular ROBPs, we let $\widetilde{\mathbf{W}}_{j\leftarrow i}$ consist of the estimated probabilities of going from vertices in layer i to vertices in layer j based on the BRRY PRG [BRRY14], which gives us bounds such as $\|\widetilde{\mathbf{W}}_{j\leftarrow i} - \mathbf{W}_{j\leftarrow i}\|_1 \leq \tau$ where τ is moderately small.

Given these matrices $\widetilde{\mathbf{W}}_{j\leftarrow i}$, a natural approach for constructing $\widehat{\mathbf{L}}^{-1}$ would be to simply arrange them in an $(n+1) \times (n+1)$ array:

$$\text{First Attempt:} \quad \widehat{\mathbf{L}}^{-1} = \begin{bmatrix} \mathbf{I} & 0 & \cdots & 0 \\ \widetilde{\mathbf{W}}_{1\leftarrow 0} & \mathbf{I} & \cdots & 0 \\ \widetilde{\mathbf{W}}_{2\leftarrow 0} & \widetilde{\mathbf{W}}_{2\leftarrow 1} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ \widetilde{\mathbf{W}}_{n\leftarrow 0} & \widetilde{\mathbf{W}}_{n\leftarrow 1} & \cdots & \mathbf{I} \end{bmatrix}. \tag{2}$$

Indeed, this approach has been used in prior work [CDRST21; Hoz21]. The trouble with this approach is that if we defined $\widehat{\mathbf{L}}^{-1}$ in this way, then the error of the matrix $\widehat{\mathbf{L}}^{-1}$ as a whole would be typically much larger than the error of a single block. For example, the condition $\|\widetilde{\mathbf{W}}_{j\leftarrow i} - \mathbf{W}_{j\leftarrow i}\|_1 \leq \tau$ would merely give us $\|\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L}\|_1 \leq O(n \cdot \tau)$, essentially due to a "union bound" over the $n+1$ rows/columns. Therefore, if we used this approach, we would have to start with a very small initial error $\tau < \frac{1}{n}$ to get a nontrivial bound ($\|\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L}\| < 1$). In the settings we study, we cannot afford to compute $\widetilde{\mathbf{W}}_{j\leftarrow i}$ with such a low error. For example, notice that all the explicit PRGs in Table 1, Table 2, and Table 3 have seed length $\Omega(\log n \cdot \log(1/\tau))$ for error τ. If used these PRGs with an initial error value of $\tau < 1/n$, then our seed lengths would always be $\Omega(\log^2 n)$. We therefore need a different approach.

3.1.3 Constructing $\widehat{\mathbf{L}}^{-1}$ via Shortcutting and Correction Graphs

Our construction of $\widehat{\mathbf{L}}^{-1}$ uses ideas from Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's algorithm for estimating random walk probabilities in Eulerian digraphs [AKMPSV20]. To briefly summarize their algorithm, they recursively applied Rozenman and Vadhan's derandomized square operation [RV05] to construct a "unit-circle approximation" with a moderate error $\tau \approx 1/\log n$, and then they decrease the error using heavy machinery from the literature on fast Laplacian solvers.

One of our contributions is to isolate and reformulate the key ingredient of their algorithm that enables them to perform error reduction starting from such a moderate error value (and avoiding the union bound over $n+1$ rows/columns). Surprisingly, this key ingredient is not the derandomized squaring or the "unit-circle approximation" but the recursive structure itself. We capture such recursive structure with the following definition of a *shortcut graph*.

For simplicity, from now on, we always assume n is a power of 2. The shortcut graph on $n+1$ vertices, denoted as SC_n, has vertex set $\{0, 1, 2 \ldots, n\}$ and contains edges $(i, i+2^k)$ for every $k \in \{0, 1, \ldots, \log n\}$ and i that is a multiple of 2^k. See Figure 1.

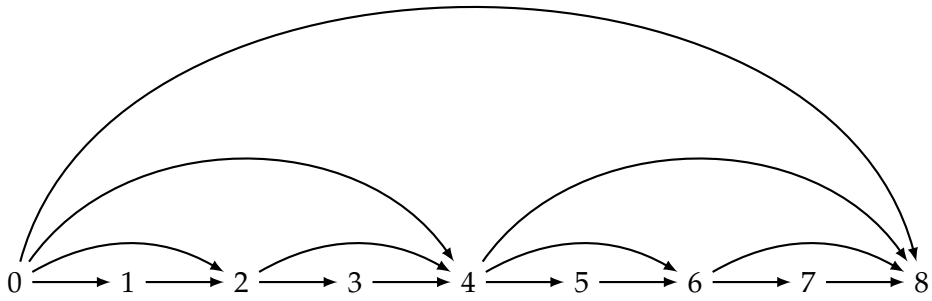

Figure 1: The shortcut graph SC_n with $n = 8$.

Construction of $\widehat{\mathbf{L}}^{-1}$ based on Shortcut Graph. Recall that for each $i < j$, we have a matrix $\widetilde{\mathbf{W}}_{j\leftarrow i}$ that we think of as a "moderate-error approximation" to $\mathbf{W}_{j\leftarrow i}$. Using these matrices $\widetilde{\mathbf{W}}_{j\leftarrow i}$, let us define the matrix $\widehat{\mathbf{L}}^{-1} \in (\mathbb{R}^{w\times w})^{(n+1)\times(n+1)}$ block-by-block. Let $0 \le \ell < r \le n$, and let $\ell = i_0 \to i_1 \to \cdots \to i_k = r$ be the (unique) *shortest path* from ℓ to r in SC_n. It is not hard to see from Figure 1 that the length of this shortest path is at most $2\log n$. We define the (r, ℓ)-th block of $\widehat{\mathbf{L}}^{-1}$ by the formula

$$(\widehat{\mathbf{L}}^{-1})_{r\leftarrow\ell} \coloneqq \widetilde{\mathbf{W}}_{i_k\leftarrow i_{k-1}} \cdot \widetilde{\mathbf{W}}_{i_{k-1}\leftarrow i_{k-2}} \cdots \widetilde{\mathbf{W}}_{i_2\leftarrow i_1} \cdot \widetilde{\mathbf{W}}_{i_1\leftarrow i_0}.$$

Intuitively, since each $\widetilde{\mathbf{W}}_{i_t\leftarrow i_{t-1}}$ approximates $\mathbf{W}_{i_t\leftarrow i_{t-1}}$, their product should approximate the product

$$\mathbf{W}_{i_k\leftarrow i_{k-1}} \cdot \mathbf{W}_{i_{k-1}\leftarrow i_{k-2}} \cdots \mathbf{W}_{i_2\leftarrow i_1} \cdot \mathbf{W}_{i_1\leftarrow i_0} = \mathbf{W}_{r\leftarrow\ell},$$

which is the (r, ℓ)-th block of the exact inverse Laplacian $\mathbf{L}^{-1}$. To complete the definition of $\widehat{\mathbf{L}}^{-1}$, we set $(\widehat{\mathbf{L}}^{-1})_{\ell\leftarrow\ell} = \mathbf{I}_w$ for every $\ell \in \{0, 1, \ldots, n\}$, and $(\widehat{\mathbf{L}}^{-1})_{r\leftarrow\ell} = \mathbf{0}$ for every $\ell > r$. For example, the case $n = 4$ is shown below.

$$\widehat{\mathbf{L}}^{-1} = \begin{bmatrix} \mathbf{I} & 0 & 0 & 0 & 0 \\ \widetilde{\mathbf{W}}_{1\leftarrow 0} & \mathbf{I} & 0 & 0 & 0 \\ \widetilde{\mathbf{W}}_{2\leftarrow 0} & \widetilde{\mathbf{W}}_{2\leftarrow 1} & \mathbf{I} & 0 & 0 \\ \widetilde{\mathbf{W}}_{3\leftarrow 2} \cdot \widetilde{\mathbf{W}}_{2\leftarrow 0} & \widetilde{\mathbf{W}}_{3\leftarrow 2} \cdot \widetilde{\mathbf{W}}_{2\leftarrow 1} & \widetilde{\mathbf{W}}_{3\leftarrow 2} & \mathbf{I} & 0 \\ \widetilde{\mathbf{W}}_{4\leftarrow 0} & \widetilde{\mathbf{W}}_{4\leftarrow 2} \cdot \widetilde{\mathbf{W}}_{2\leftarrow 1} & \widetilde{\mathbf{W}}_{4\leftarrow 2} & \widetilde{\mathbf{W}}_{4\leftarrow 3} & \mathbf{I} \end{bmatrix}. \tag{3}$$

Comparisons. Our construction of $\widehat{\mathbf{L}}^{-1}$ based on the shortcut graph corresponds to Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's recursive usage of the Schur complement operation [AKMPSV20, Theorem 6.1].

Let us compare our definition of $\widehat{\mathbf{L}}^{-1}$ above to the straightforward approach given in (2). The straightforward approach uses "unrelated" approximations for all $\Theta(n^2)$ blocks of $\mathbf{L}^{-1}$. Instead, we first approximate $O(n)$ many (carefully selected) "essential" blocks $\mathbf{W}_{j\leftarrow i}$, and then we approximate each remaining block by multiplying approximations of essential blocks. The

"price" we pay for this additional structure is low, because each product involves only $O(\log n)$ many essential blocks.

Analyzing $\widehat{\mathbf{L}}^{-1}$: Inverse analysis of random walks. Having defined $\widehat{\mathbf{L}}^{-1}$, our job is to bound $\|\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L}\|$, i.e., our job is to show that $\widehat{\mathbf{L}}^{-1}$ approximately describes random walks in the given branching program B in a certain sense. At an intuitive level, our approach is as follows. We admit that $\widehat{\mathbf{L}}^{-1}$ doesn't *perfectly* describe random walks in B; we only aim to show that it is an approximation. However, $\widehat{\mathbf{L}}^{-1}$ *does* perfectly describe random walks in *some other* graph $\widehat{B}$! This graph $\widehat{B}$ is not a true ROBP (it is not layered, and its edges have positive and negative weights), but still, our construction of $\widehat{\mathbf{L}}^{-1}$ ensures that $\widehat{B}$ has considerable combinatorial structure. We use this structure to show that $\widehat{B} \approx B$ in some sense, which enables us to bound the error $\|\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L}\|$. The process of going from $\widehat{\mathbf{L}}^{-1}$ to $\widehat{B}$ (i.e., starting with a description of the behavior of random walks, and reconstructing an underlying graph that is consistent with that description) is what we refer to as the "inverse analysis of random walks." [8]

The matrix $\widehat{\mathbf{L}}$ and the correction graph. To implement the plan described above, let $\widehat{\mathbf{L}}$ denote the inverse of $\widehat{\mathbf{L}}^{-1}$; this exists because $\widehat{\mathbf{L}}^{-1}$ is lower-triangular. We emphasize that we first construct an approximate inverse to $\mathbf{L}$ (which we denote by $\widehat{\mathbf{L}}^{-1}$), and then we take the *exact* inverse of *that* matrix to get $\widehat{\mathbf{L}}$. Then $\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L} = \widehat{\mathbf{L}}^{-1} \cdot (\widehat{\mathbf{L}} - \mathbf{L})$, so our job is to bound $\|\widehat{\mathbf{L}}^{-1} \cdot (\widehat{\mathbf{L}} - \mathbf{L})\|$. Actually, we will show that it suffices to bound $\|\mathbf{L}^{-1} \cdot (\widehat{\mathbf{L}} - \mathbf{L})\|$:

$$\text{Suppose } \|\mathbf{L}^{-1} \cdot (\widehat{\mathbf{L}} - \mathbf{L})\| \le \delta. \quad \text{Then } \|\widehat{\mathbf{L}}^{-1} \cdot (\widehat{\mathbf{L}} - \mathbf{L})\| \le \delta/(1-\delta). \qquad \text{(Corollary 4.11)}$$

We will therefore focus on bounding $\|\mathbf{L}^{-1}(\widehat{\mathbf{L}} - \mathbf{L})\|$ instead.

It turns out that this matrix $\widehat{\mathbf{L}}$ is highly structured. For example, it is relatively *sparse*: we show that the block $\widehat{\mathbf{L}}_{j \leftarrow i}$ (where $i < j$) is nonzero only if $(i,j) \in E(\mathsf{SC}_n)$. Furthermore, we give an exact formula for $\widehat{\mathbf{L}}$. Assume for simplicity that all length-1 approximations are exact, i.e., $\widetilde{\mathbf{W}}_{i+1 \leftarrow i} = \mathbf{W}_{i+1 \leftarrow i}$. We show (Lemma 4.5) that

$$\widehat{\mathbf{L}} = \mathbf{I} - (\mathbf{W} + \Delta\mathbf{W}),$$

where $\Delta\mathbf{W} = \sum_{t=1}^{\log n} \Delta\mathbf{W}^{(t)}$ and

$$(\Delta\mathbf{W}^{(t)})_{j \leftarrow i} = \begin{cases} \widetilde{\mathbf{W}}_{j \leftarrow i} - \widetilde{\mathbf{W}}_{j \leftarrow \frac{i+j}{2}} \widetilde{\mathbf{W}}_{\frac{i+j}{2} \leftarrow i} & \text{if } (i,j) \in E(\mathsf{SC}_n) \text{ and } j = i + 2^t \\ 0 & \text{otherwise.} \end{cases}$$

For example, the inverse of the $n = 4$ example from (3) is given below:

$$\widehat{\mathbf{L}} = \begin{bmatrix} \mathbf{I} & 0 & 0 & 0 & 0 \\ -\widetilde{\mathbf{W}}_{1 \leftarrow 0} & \mathbf{I} & 0 & 0 & 0 \\ -(\widetilde{\mathbf{W}}_{2 \leftarrow 0} - \widetilde{\mathbf{W}}_{2 \leftarrow 1}\widetilde{\mathbf{W}}_{1 \leftarrow 0}) & -\widetilde{\mathbf{W}}_{2 \leftarrow 1} & \mathbf{I} & 0 & 0 \\ 0 & 0 & -\widetilde{\mathbf{W}}_{3 \leftarrow 2} & \mathbf{I} & 0 \\ -(\widetilde{\mathbf{W}}_{4 \leftarrow 0} - \widetilde{\mathbf{W}}_{4 \leftarrow 2}\widetilde{\mathbf{W}}_{2 \leftarrow 0}) & 0 & -(\widetilde{\mathbf{W}}_{4 \leftarrow 2} - \widetilde{\mathbf{W}}_{4 \leftarrow 3}\widetilde{\mathbf{W}}_{3 \leftarrow 2}) & -\widetilde{\mathbf{W}}_{4 \leftarrow 3} & \mathbf{I} \end{bmatrix}.$$

Since $\widehat{\mathbf{L}} = \mathbf{I} - (\mathbf{W} + \Delta\mathbf{W})$, the matrix $\widehat{\mathbf{L}}$ can be interpreted as the "Laplacian" of a weighted graph with transition matrix $\mathbf{W} + \Delta\mathbf{W}$. We refer to the subgraph corresponding to $\Delta\mathbf{W}$ as the "Correction Graph," because adding $\Delta\mathbf{W}$ to $\widehat{\mathbf{L}}$ yields the original Laplacian $\mathbf{L}$.

[8] One caveat is that because of the positive and negative edge weights in $\widehat{B}$, technically we ought to speak of "weighted walks" rather than "random walks;" we will ignore this distinction for simplicity.

Overall, we have

$$\|\mathbf{L}^{-1} \cdot (\widehat{\mathbf{L}} - \mathbf{L})\| = \|\mathbf{L}^{-1}\Delta\mathbf{W}\| \leq \sum_{t=1}^{\log n} \|\mathbf{L}^{-1}\Delta\mathbf{W}^{(t)}\|,$$

and so our job reduces to bounding $\|\mathbf{L}^{-1}\Delta\mathbf{W}^{(t)}\|$ for a fixed $t \in [\log n]$.

In summary, for a standard-order ROBP B, we have the following recipe for estimating $\mathbb{E}[B]$ to within low error.

1. Design an initial algorithm for constructing the approximation matrices $\widetilde{\mathbf{W}}_{j\leftarrow i}$. These matrices $\widetilde{\mathbf{W}}_{j\leftarrow i}$ induce a correction graph transition matrix $\Delta\mathbf{W}$.
2. Pick a submultiplicative matrix norm (or "extended seminorm") $\|\cdot\|$. In this paper, the function $\|\cdot\|$ is always induced by a vector seminorm (see Definition 2.4).
3. Use properties of the branching program B (e.g., regularity) to prove that $\|\mathbf{L}^{-1}\Delta\mathbf{W}^{(t)}\| \leq \delta$ for each $t \in [\log n]$, where $\mathbf{L}$ is the Laplacian matrix of B and δ is "moderately small."
4. Apply our error reduction framework (the construction of $\widehat{\mathbf{L}}^{-1}$, the correction graph lemma, Richardson iteration, etc.) to obtain a matrix $\mathbf{A}_m$ that has very low error, in the sense that $\|\mathbf{I} - \mathbf{A}_m\mathbf{L}\|$ is very small.
5. Use $\mathbf{A}_m$ to compute an approximation to $\mathbb{E}[B]$, and use properties of the "norm function" $\|\cdot\|$ to conclude that this approximation has very small additive error.

The details of how we carry out this recipe to prove our main results vary from one result to the next. In the remainder of this technical overview, we give an overview of each of the arguments.

3.2 Our WPRG for Bounded-Width Regular ROBPs

In this section, we explain how we carry out each step of the recipe to construct a WPRG that ε-fools bounded-width regular ROBPs (Theorem 1.7). For ease of exposition, we will focus on the case $\varepsilon = 1/n$ in this proof overview, and we will assume that the width of the program is $O(1)$.

We construct the approximation matrices $\widetilde{\mathbf{W}}_{j\leftarrow i}$ by using the BRRY PRG [BRRY14]. We set the error parameter of the PRG to $\tau = 2^{-\sqrt{\log n}}$, so the BRRY PRG has seed length $\widetilde{O}(\log n \cdot \log(1/\tau)) = \widetilde{O}(\log^{3/2} n)$. We use the standard ℓ_∞ matrix norm, i.e.,

$$\|\mathbf{M}\| = \max_v \sum_u |\mathbf{M}_{v,u}|.$$

Our main job is to bound $\|\mathbf{L}^{-1}\Delta\mathbf{W}^{(t)}\|$. For simplicity, let us focus on the case $t = 1$. Working through the definitions, each nonzero block of $\mathbf{L}^{-1}\Delta\mathbf{W}^{(1)}$ has the form

$$\mathbf{W}_{j\leftarrow\ell+2} \cdot (\widetilde{\mathbf{W}}_{\ell+2\leftarrow\ell} - \mathbf{W}_{\ell+2\leftarrow\ell}).$$

We analyze the matrix above using the *weight* measure introduced by Braverman, Rao, Raz, and Yehudayoff [BRRY14]. Using techniques similar to Braverman, Rao, Raz, and Yehudayoff's arguments [BRRY14], one can show (Lemma 6.6) that for any vertex u in layer ℓ and any vertex v in layer j, we have

$$|(\mathbf{W}_{j\leftarrow\ell+2} \cdot (\widetilde{\mathbf{W}}_{\ell+2\leftarrow\ell} - \mathbf{W}_{\ell+2\leftarrow\ell}))_{v,u}| \leq O(\tau \cdot \mathsf{Weight}(B^{v\leftarrow}, \ell, \ell+2)),$$

where $\mathsf{Weight}(B^{v\leftarrow}, \ell, \ell+2)$ is the sum, over all edges (u, u') between layer ℓ and layer $\ell+2$, of $|\mathbb{E}[B^{v\leftarrow u}] - \mathbb{E}[B^{v\leftarrow u'}]|$. Therefore, summing over all ℓ, we get

$$\sum_u |(\mathbf{L}^{-1}\Delta\mathbf{W}^{(1)})_{v,u}| \le O(\tau \cdot \mathsf{Weight}(B^{v\leftarrow}, 0, n)).$$

Braverman, Rao, Raz, and Yehudayoff showed that because B is regular, we have $\mathsf{Weight}(B^{v\leftarrow}, 0, n) = O(1)$, and hence $\|\mathbf{L}^{-1}\Delta\mathbf{W}^{(1)}\| \le O(\tau)$.

A similar argument bounds $\|\mathbf{L}^{-1}\Delta\mathbf{W}^{(t)}\|$ when $t > 1$. From here, our error reduction framework provides a matrix $\mathbf{A}_m$ such that $\|\mathbf{I} - \mathbf{A}_m\mathbf{L}\| \le \Theta(1/n)$. By reverse-engineering the definition of $\mathbf{A}_m$, we construct a WPRG that fools B with error $1/n$ as desired. However, the seed length of this WPRG is too large, because each seed includes several independent seeds for the BRRY PRG. We therefore decrease the seed length by using the Impagliazzo-Nisan-Wigderson (INW) PRG [INW94] to generate *correlated* seeds for the BRRY PRG, similar to prior work [CDRST21; PV21]. See Section 6 for details.

3.3 Our WPRG for Width-3 ROBPs

Next, we give an overview of our WPRG for width-3 ROBPs (Theorem 1.5). Recall that Meka, Reingold, and Tal designed a PRG for width-3 length-n standard-order ROBPs with seed length $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$ [MRT19]. We do not apply our error reduction framework to this PRG in a black-box way. Instead, to construct our WPRG, we revisit Meka, Reingold, and Tal's specific construction and analysis [MRT19].

To construct their PRG, Meka, Reingold, and Tal showed how to sample a pseudorandom *restriction* $\rho \in \{0, 1, \star\}^n$ using $\widetilde{O}(\log(n/\varepsilon))$ truly random bits such that the following two properties hold for any width-3 length-n standard-order ROBP B.

1. The restriction ρ *preserves the expectation* of B, i.e., $|\mathbb{E}_{\rho,\mathcal{U}}[B|_\rho(\mathcal{U})] - \mathbb{E}[B]| \le \varepsilon$, where $\mathcal{U}$ is independent of ρ and uniform random.

2. The program B *simplifies* under ρ with high probability. Indeed, it "almost" becomes a permutation ROBP.

In more detail, with probability $1-\varepsilon$, the restricted program $B|_\rho$ is approximated by a program $\widetilde{B}$ that satisfies the permutation condition (Definition 1.8) in all but a few layers. In this case, Meka, Reingold, and Tal argue that the BRRY PRG [BRRY14] fools $B|_\rho$ [MRT19]. Therefore, to fool the original program B, they apply the pseudorandom restriction ρ and then fill in the stars using the BRRY PRG. The overall seed length is dominated by the $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$ seed length of the BRRY PRG.

Our WPRG for width-3 ROBPs follows the same high-level plan, except that in the final step, instead of applying the BRRY PRG, we apply our own WPRG for regular ROBPs. That way, instead of paying $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$ truly random bits in the final step, we only pay $\widetilde{O}\left(\log n \cdot \sqrt{\log(1/\varepsilon)} + \log(1/\varepsilon)\right)$ truly random bits. Implementing this plan requires improving Meka, Reingold, and Tal's "simplification" arguments in multiple ways, two of which we describe below.

A quantitative improvement: Fewer colliding layers. In Meka, Reingold, and Tal's analysis, the number of "colliding layers" in $\widetilde{B}$ (i.e., layers that violate the permutation condition) is bounded by $\mathrm{poly}(1/\varepsilon) \cdot \log\log n$ [MRT19]. We are most interested in the regime $\varepsilon = 1/\mathrm{poly}(n)$, in which their

bound is trivial. To address this issue, we show that with probability $1-\varepsilon$, the restricted program $B|_\rho$ is in fact approximated by a program with only $\mathrm{polylog}(1/\varepsilon) \cdot \log\log n$ many colliding layers (see Claim 7.35).

A qualitative improvement: Bounding the weight of the restricted program. Building on our analysis of colliding layers, we show that with high probability, the restricted function $B|_\rho$ can be computed by a program with bounded *weight* (as defined by Braverman, Rao, Raz, and Yehudayoff [BRRY14]). Specifically, with probability $1-\varepsilon$, the weight is at most $\mathrm{polylog}(n/\varepsilon)$; see Theorem 7.2.

We consider this "simplification" statement in terms of weight to be cleaner and easier to understand, compared to Meka, Reingold, and Tal's analysis [MRT19]. Furthermore, it turns out to be a crucial step in our WPRG analysis. To explain why, let $\mathcal{X}$ be a pseudorandom string sampled by the BRRY PRG [BRRY14]. As mentioned previously, Meka, Reingold, and Tal prove that $\mathcal{X}$ fools the restricted program $B|_\rho$ [MRT19]. Let us recall their argument in more detail, so that we can see where it breaks down when we try to use our WPRG in place of $\mathcal{X}$.

The first step of the argument is to show that $\mathcal{X}$ fools programs with few colliding layers such as $\widetilde{B}$ [MRT19]. Then, to bridge the gap between $\widetilde{B}$ and $B|_\rho$, the second step is to design an "error indicator" function E with the following properties:

1. For every x such that $\widetilde{B}(x) \neq B|_\rho(x)$, we have $E(x) = 1$.
2. $\mathbb{E}[E] \leq \varepsilon$ and $\mathbb{E}[E(\mathcal{X})] \leq \varepsilon$.

Because such an E exists, Meka, Reingold, and Tal are able to reason that

$$\begin{aligned} \left|\mathbb{E}[B|_\rho] - \mathbb{E}\left[\widetilde{B}\right]\right| &\leq \mathbb{E}[E] \leq \varepsilon \\ \text{and } \left|\mathbb{E}[B|_\rho(\mathcal{X})] - \mathbb{E}\left[\widetilde{B}(\mathcal{X})\right]\right| &\leq \mathbb{E}[E(\mathcal{X})] \leq \varepsilon, \end{aligned} \tag{4}$$

and consequently $\mathbb{E}[B|_\rho(\mathcal{X})] \approx \mathbb{E}\left[\widetilde{B}(\mathcal{X})\right] \approx \mathbb{E}\left[\widetilde{B}\right] \approx \mathbb{E}[B|_\rho]$. (This argument can be interpreted as a type of "sandwiching argument.")

The preceding argument breaks down when we replace the BRRY PRG with our WPRG $(\mathcal{G}, \mu)$. To understand the issue, let S_{err} be the set of seeds x such that $\widetilde{B}(\mathcal{G}(x)) \neq B|_\rho(\mathcal{G}(x))$, and let S_E be the set of seeds x such that $E(\mathcal{G}(x)) = 1$. Then $S_{\mathrm{err}} \subseteq S_E$, but we might nevertheless have

$$\left|\sum_{x \in S_{\mathrm{err}}} \mu(x)\right| \gg \left|\sum_{x \in S_E} \mu(x)\right|$$

due to cancellations in the right-hand sum. Therefore, (4) no longer works after we introduce negative weights. (In general, WPRGs do not seem to be "compatible" with sandwiching arguments.)

We circumvent this issue by eliminating the error indicator function altogether. In a nutshell, this is possible due to the following facts: (i) The program $\widetilde{B}$ is a "suffix" of the restricted program $B|_\rho$, i.e., it consists of layers $i, i+1, \ldots, n$ of $B|_\rho$ for some i. (ii) The error indicator function $E(x)$ essentially checks whether all paths that start in layer i of $B|_\rho$ collide under the input x. (iii) If these paths collide with high probability (under a uniformly random input x), then the weights in $B|_\rho$ *before* layer i are negligible. (iv) Since $\widetilde{B}$ has few colliding layers, the total weight of the edges in $B|_\rho$ *after* layer i are also bounded. Thus, we are able to bound the total weight of $B|_\rho$.

In the analysis of our WPRG for regular ROBPs, the only place we use regularity is to bound the weight of the program (and its subprograms[9]). Therefore, since we show that $B|_\rho$ can be computed by a program with low weight, it follows that our WPRG fools $B|_\rho$. By combining with the pseudorandom restriction ρ, we get a WPRG that fools B itself. See Section 7 for details.

3.4 Our Simplified Derandomization of Polynomial-Width Regular ROBPs

We now present an overview of our simplified proof of Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's main result [AKMPSV20] (Theorem 1.10). That is, given the description of a regular width-w length-n standard-order ROBP B, we will explain how to estimate $\mathbb{E}[B]$ to within a small additive error in near-logarithmic space. The most important case to keep in mind is $w = \mathrm{poly}(n)$.

Our proof is largely inspired by Braverman, Rao, Raz, and Yehudayoff's work [BRRY14]. To explain the intuition, let us begin by revisiting the analysis of *constant-width* regular ROBPs that we summarized in Section 3.2, so we can understand more deeply why it works. Afterward, we will explain how to adapt the intuition to the polynomial-width case.

3.4.1 Reflections on Braverman, Rao, Raz, and Yehudayoff's Techniques [BRRY14]

Let $V^{(0)}, \dots, V^{(n)}$ be the layers of a constant-width regular program B. Fix some target vertex v, say $v \in V^{(j)}$. To show that $\|\mathbf{L}^{-1}\Delta\mathbf{W}^{(1)}\|_\infty \le O(\tau)$, where τ is the error of each block $\widetilde{\mathbf{W}}_{r\leftarrow\ell}$, the main steps that we discussed in Section 3.2 are as follows.

1. First, we *bound the "local" error in terms of weight*. Let $u \in V^{(\ell)}$ where $\ell \le j-2$. We show
$$|(\mathbf{W}_{j\leftarrow\ell+2} \cdot (\widetilde{\mathbf{W}}_{\ell+2\leftarrow\ell} - \mathbf{W}_{\ell+2\leftarrow\ell}))_{v,u}| \le O(\tau \cdot \mathsf{Weight}(B^{v\leftarrow}, \ell, \ell+2)).$$
2. Then, we *bound the total weight of the program*. Braverman, Rao, Raz, and Yehudayoff showed that $\mathsf{Weight}(B^{v\leftarrow}, 0, n) \le O(1)$ [BRRY14]. Consequently, the total error is $O(\tau)$.

To gain more insight, let us open up the two proofs above and look inside.

The BRRY potential argument. Let S be the set of vertices $q \in V^{(\ell+2)}$ that are reachable from u. (See Figure 2.) Define
$$\mathrm{Spread}(u) := \left(\max_{q\in S} \mathbb{E}[B^{v\leftarrow q}]\right) - \left(\min_{q\in S} \mathbb{E}[B^{v\leftarrow q}]\right),$$
and consider two extreme cases:

1. **(The error-free case.)** Suppose $\mathrm{Spread}(u) = 0$, i.e., $\mathbb{E}[B^{v\leftarrow q}]$ is the same for every $q \in S$. In this case, the two bits that we read immediately after visiting u "do not matter." Consequently, there is no error: $|(\mathbf{W}_{j\leftarrow\ell+2} \cdot (\widetilde{\mathbf{W}}_{\ell+2\leftarrow\ell} - \mathbf{W}_{\ell+2\leftarrow\ell}))_{v,u}| = 0$. In general, one can show that the error $|(\mathbf{W}_{j\leftarrow\ell+2} \cdot (\widetilde{\mathbf{W}}_{\ell+2\leftarrow\ell} - \mathbf{W}_{\ell+2\leftarrow\ell}))_{v,u}|$ is bounded by $O(\tau \cdot \mathrm{Spread}(u))$.
2. **(The mixing case.)** Suppose $\mathrm{Spread}(u) \gg 0$. In this case, it is helpful to think about the *reversed random walk*, in which we reverse all the directions of the edges in B. The assumption $\mathrm{Spread}(u) \gg 0$ means that there are two vertices in $V^{(\ell+2)}$ that have very

[9] Because of this technicality, we must argue that not only does $B|_\rho$ have low weight, but also all of its subprograms have low weight. See Theorem 7.2.

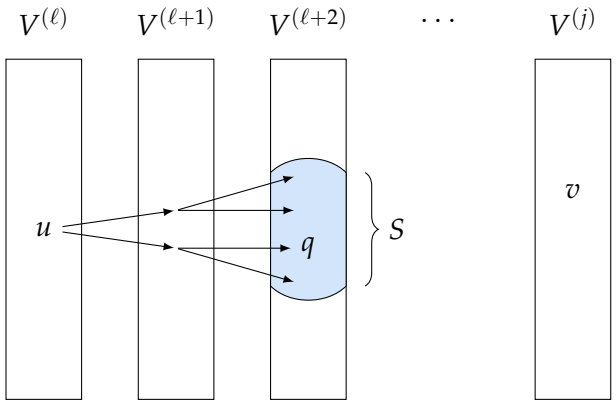

Figure 2: The setup for the analysis of constant-width regular ROBPs.

different probabilities of being reached *from* v, and u is reachable from both of them. Intuitively, because B is regular, this should mean that in the reversed random walk, the distribution over states becomes *more uniform* (mixed) when we walk from $V^{(\ell+2)}$ to $V^{(\ell)}$.

The ideas above enable us to carry out a *potential argument*. Define a potential function by the formula

$$\Phi_i := \sum_{a,a' \in V^{(i)}} |\mathbb{E}[B^{v \leftarrow a}] - \mathbb{E}[B^{v \leftarrow a'}]|,$$

which quantifies how far the reversed random walk is from uniform ("fully mixed") when it reaches $V^{(i)}$ [BRRY14]. For each $u \in V^{(\ell)}$, one can use the concept of "weight" to prove

$$\Phi_{\ell+2} \geq \Phi_\ell + \Omega\left(\mathrm{Spread}(u)\right).$$

Therefore, whenever there is any error, there is a corresponding increase in the potential function Φ_i. Meanwhile, the potential function's total growth is bounded, i.e., $\Phi_n \leq O(1)$. Therefore, the sum of errors over all vertices u (which corresponds to $\|\mathbf{L}^{-1}\Delta\mathbf{W}^{(1)}\|_\infty$) must be bounded by $O(\tau)$.

Linear-algebraic interpretation. We can interpret the quantity $\mathrm{Spread}(u)$ in linear-algebraic terms as follows. Let $D = 4$ be the number of paths from u to $V^{(\ell+2)}$. If the i-th path from u leads to $q \in V^{(r)}$, then define

$$y_i^{(u)} = \mathbb{E}[B^{v \leftarrow q}].$$

Thus, $y^{(u)}$ is a vector in $\mathbb{R}^D$. Decompose $y^{(u)} = y_\parallel^{(u)} + y_\perp^{(u)}$, where $y_\parallel^{(u)}$ is parallel to the all-ones vector and $y_\perp^{(u)}$ is perpendicular to the all-ones vector. Intuitively, the component $y_\parallel^{(u)}$ corresponds to the "error-free" case, and $y_\perp^{(u)}$ corresponds to the "mixing" case. Furthermore,

$$\|y_\perp^{(u)}\|_\infty \leq \mathrm{Spread}(u) \leq 2 \cdot \|y_\perp^{(u)}\|_\infty.$$

Thus, by looking at $\mathrm{Spread}(u)$, we are effectively using the ℓ_∞ norm of $y_\perp^{(u)}$ to tell us whether we are in Case 1 ("error-free") or Case 2 ("mixing").

Key new idea: Using the ℓ_2 norm. Why do we use the ℓ_∞ norm to measure $y_\perp^{(u)}$? Intuitively, the underlying reason is that we are working with an ℓ_∞-type guarantee regarding the approximation matrices $\widetilde{\mathbf{W}}_{j\leftarrow i}$. To get the best WPRG seed length, we construct $\widetilde{\mathbf{W}}_{j\leftarrow i}$ using the BRRY PRG [BRRY14] (as discussed in Section 3.2), but the correctness proof works in the more general setting where $\widetilde{\mathbf{W}}_{j\leftarrow i}$ is constructed using an *arbitrary* PRG that fools constant-width regular ROBPs with moderate error. In such a general setting, using the ℓ_∞ norm seems necessary, because it captures the "worst possible error" for a pseudorandom walk starting from u.

To make progress in the polynomial-width case, we take a less "generic" approach. We construct the approximation matrices $\widetilde{\mathbf{W}}_{j\leftarrow i}$ by recursively applying (a version of) Rozenman and Vadhan's derandomized squaring operation [RV05] (similar to what Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan did [AKMPSV20]). This operation uses a *spectral expander graph* to recycle randomness. Let $\mathbf{H}$ be the expander's transition matrix, and let $\mathbf{J}$ be the transition matrix of a complete graph with self-loops. The quality of $\mathbf{H}$ is measured by the quantity $\lambda(\mathbf{H}) = \|\mathbf{H} - \mathbf{J}\|_2$, i.e., we measure error using the ℓ_2 norm. This raises the following question:

> In the case that the matrices $\widetilde{\mathbf{W}}_{j\leftarrow i}$ are constructed using the derandomized square, can we get a tighter analysis of $\mathbf{L}^{-1}\Delta\mathbf{W}$ by looking at the ℓ_2 norm $\|y_\perp^{(u)}\|_2$ instead of the ℓ_∞ norm?

Although seemingly naïve, this idea is the core motivation for our simplified proof of Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's main result [AKMPSV20]. We believe it is the "real magic" behind their work.

3.4.2 The Error-Free Case: Singular-Value Approximation

Let B be a regular width-w length-n standard-order ROBP, where $w = \text{poly}(n)$. As discussed above, our approach for analyzing $\mathbf{L}^{-1}\Delta\mathbf{W}$ is to use the ℓ_2 norm to distinguish between an "error-free" case and a "mixing" case. We begin by discussing the error-free case.

For simplicity, we continue to focus on $\Delta\mathbf{W}^{(1)}$ for this proof overview. Therefore, fix a length-2 edge $(\ell, \ell+2) \in E(\mathsf{SC}_n)$. The approximation matrix $\widetilde{\mathbf{W}}_{\ell+2\leftarrow\ell}$ is defined by using an expander graph to recycle randomness. Let $\mathbf{H} \in \mathbb{R}^{d\times d}$ be the transition matrix of this expander, and let $\mathbf{J} \in \mathbb{R}^{d\times d}$ be the transition matrix of the complete graph with self-loops. The definition of $\lambda(\mathbf{H})$ implies, for all $\alpha, \beta \in \mathbb{R}^d$,

$$\left|\beta^T \cdot (\mathbf{H} - \mathbf{J}) \cdot \alpha\right| \le \lambda(\mathbf{H}) \cdot \|\alpha\|_2 \cdot \|\beta\|_2 \le \frac{\lambda(\mathbf{H})}{2} \cdot (\|\alpha\|_2^2 + \|\beta\|_2^2). \tag{5}$$

Now let $x \in \mathbb{R}^{V^{(\ell)}}$ and $y \in \mathbb{R}^{V^{(\ell+2)}}$ be vectors. For each vertex $u \in V^{(\ell)}$, let $y^{(u)}$ be the subvector of y that is reachable from u. That is, if the i-th path from u leads to v, then we define $y_i^{(u)} = y_v$. Similarly, for each vertex $v \in V^{(\ell+2)}$, let $x^{(v)}$ be the subvector of x from which v is reachable. By regularity, $x^{(v)}$ and $y^{(u)}$ are both vectors in $\mathbb{R}^D$ where $D = 2^2 = 4$. Using (5), one can show that

$$\left|y^T \cdot \left(\widetilde{\mathbf{W}}_{\ell+2\leftarrow\ell} - \mathbf{W}_{\ell+2\leftarrow\ell}\right) \cdot x\right| \le \frac{\lambda(\mathbf{H})}{2D} \cdot \left(\sum_{v\in V^{(\ell+2)}} \|x_\perp^{(v)}\|_2^2 + \sum_{u\in V^{(\ell)}} \|y_\perp^{(u)}\|_2^2\right). \tag{6}$$

(See Lemma A.15 and the proof of Corollary A.16.) When x and y are chosen appropriately, the left-hand side of (6) measures the amount of error that occurs in layers ℓ through $\ell+2$. Therefore, provided that all of the ℓ_2 norms $\|x_\perp^{(v)}\|_2$ and $\|y_\perp^{(u)}\|_2$ are close to zero, (6) says that we are in an "error-free" case, as desired.

The sums on the right-hand side of (6) can be conveniently simplified as follows. Observe that $\|x_\perp^{(v)}\|_2^2 = \|x^{(v)}\|_2^2 - \|x_\parallel^{(v)}\|_2^2$. Intuitively, $x_\parallel^{(v)}$ corresponds to the v-th coordinate of $\mathbf{W}_{\ell+2\leftarrow\ell} \cdot x$. Using this idea, one can show that

$$\frac{1}{D} \cdot \sum_{v \in V^{(r)}} \|x_\perp^{(v)}\|_2^2 = \|x\|_2^2 - \|\mathbf{W}_{\ell+2\leftarrow\ell} \cdot x\|_2^2 = \|x\|^2_{\mathbf{I}-\mathbf{W}_{\ell+2\leftarrow\ell}^T \mathbf{W}_{\ell+2\leftarrow\ell}},$$

and similarly,

$$\frac{1}{D} \cdot \sum_{u \in V^{(\ell)}} \|y_\perp^{(u)}\|_2^2 = \|y\|_2^2 - \|\mathbf{W}_{\ell+2\leftarrow\ell}^T \cdot y\|_2^2 = \|y\|^2_{\mathbf{I}-\mathbf{W}_{\ell+2\leftarrow\ell}\mathbf{W}_{\ell+2\leftarrow\ell}^T}.$$

Thus, (6) is equivalent to the statement that $\widetilde{\mathbf{W}}_{\ell+2\leftarrow\ell}$ is a good *singular-value approximation* of $\mathbf{W}_{\ell+2\leftarrow\ell}$, as defined by Ahmadinejad, Peebles, Pyne, Sidford, and Vadhan [APPSV23]:

Definition 3.1 (Singular-value approximation [APPSV23])**.** *Let $\widetilde{\mathbf{W}}, \mathbf{W} \in \mathbb{R}^{w \times w}$ be doubly stochastic matrices. We say $\widetilde{\mathbf{W}}$ τ-*singular-value approximates *$\mathbf{W}$, denoted as $\widetilde{\mathbf{W}} \overset{\text{sv}}{\approx}_\tau \mathbf{W}$, if for every $x, y \in \mathbb{R}^w$,*

$$\left| y^T \left(\widetilde{\mathbf{W}} - \mathbf{W} \right) x \right| \le \frac{\tau}{4} \cdot \left(\|x\|^2_{\mathbf{I}-\mathbf{W}^T\mathbf{W}} + \|y\|^2_{\mathbf{I}-\mathbf{W}\mathbf{W}^T} \right).$$

In this proof overview, we are focusing on the length-two edges in SC_n, which correspond to a single application of the derandomized squaring operation. In the actual proof, we show that more generally, $\widetilde{\mathbf{W}}_{r\leftarrow\ell}$ is a good singular-value approximation of $\mathbf{W}_{r\leftarrow\ell}$ for every edge $(\ell, r) \in E(\mathsf{SC}_n)$. (See Theorem 8.1.) We would like to clarify that it was already known that recursive derandomized squaring produces a good singular-value approximation. Indeed, this fact readily follows from the analysis in Ahmadinejad, Peebles, Pyne, Sidford, and Vadhan's recent work [APPSV23]. For this reason, we defer our alternative proof of this fact to Appendix A.

3.4.3 The Mixing Case: Potential Dynamics

To summarize the discussion so far, our simplified proof of Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's main theorem [AKMPSV20] can be divided into two main parts.

1. First, we show that the approximation matrices $\widetilde{\mathbf{W}}_{j\leftarrow i}$ constructed via a version of the derandomized squaring operation [RV05] are singular-value approximations of the corresponding exact matrices $\mathbf{W}_{j\leftarrow i}$. (See Theorem 8.1.)

2. Second, we show that $\|\mathbf{L}^{-1}\Delta\mathbf{W}\|$ is moderately small, where $\|\cdot\|$ is a suitably chosen matrix "norm."[10] (See Lemma 8.5.)

Bounding $\|\mathbf{L}^{-1}\Delta\mathbf{W}\|$ enables us to apply our error reduction framework, which ultimately leads to a low-error approximation to $\mathbb{E}[B]$. (See Theorem 8.3.)

We now present an overview of our proof that $\|\mathbf{L}^{-1}\Delta\mathbf{W}\|$ is moderately small. For this proof, the only feature of the approximation matrices $\widetilde{\mathbf{W}}_{j\leftarrow i}$ that we use is the fact that $\widetilde{\mathbf{W}}_{j\leftarrow i} \overset{\text{sv}}{\approx}_{\tau_0} \mathbf{W}_{j\leftarrow i}$ for a suitable $\tau_0 = 1/\text{polylog}(n)$.

[10] Technically, the "norm" we use is not a true norm, but rather an extended submultiplicative matrix seminorm. See Definition 2.3.

The test vector and the error vectors. Like before, we decompose $\Delta \mathbf{W} = \sum_{t=1}^{\log n} \Delta \mathbf{W}^{(t)}$. For simplicity, in this technical overview, we continue to focus on the case $t = 1$, i.e., we focus on the challenge of bounding $\|\mathbf{L}^{-1}\Delta \mathbf{W}^{(1)}\|$.

Our matrix "norm" $\|\cdot\|$ is induced by a suitable vector "norm" (technically a seminorm), which we will specify later. Therefore, fix some arbitrary "test vector" $x \in \mathbb{R}^{(n+1)\cdot w}$, and let $z = \mathbf{L}^{-1}\Delta \mathbf{W}^{(1)}x$. Our job is to show that $\|z\| \le \delta\|x\|$ for some moderately small δ.

Write x as a block vector $x = (x^{[0]}, \ldots, x^{[n]})$, where $x^{[j]} \in \mathbb{R}^{V^{(j)}} \cong \mathbb{R}^w$. Similarly, write $z = (z^{[0]}, \ldots, z^{[n]})$. Observe that we have the following recursive formula for $z^{[j]}$: for every $j \in \{0, 2, 4, \ldots, n-2\}$, we have

$$z^{[j+2]} = \mathbf{W}_{j+2\leftarrow j} \cdot z^{[j]} + \Delta \mathbf{W}^{(1)}_{j+2\leftarrow j} \cdot x^{[j]}.$$

For convenience, define $z^{[n+2]}$ to be the zero vector, and define $\mathbf{W}_{n+2\leftarrow n}$ and $\Delta \mathbf{W}^{(1)}_{n+2\leftarrow n}$ to be the zero matrix, so the equation above holds even for $j = n$. We denote the two terms above by $\widetilde{z}^{[j+2]} := \mathbf{W}_{j+2\leftarrow j} \cdot z^{[j]}$ and $s^{[j+2]} := \Delta \mathbf{W}^{(1)}_{j+2\leftarrow j} \cdot x^{[j]}$, so that

$$z^{[j+2]} = \widetilde{z}^{[j+2]} + s^{[j+2]}.$$

Intuitively, $z^{[j+2]}$ is the "cumulative" error vector at layer $j+2$, while $s^{[j+2]}$ is the "local" error vector.

The potential function. Analogous to the discussion in Section 3.4.1, our approach for bounding $\|z\|$ is to use a potential argument. Our potential function $\Phi\colon \mathbb{R}^w \to \mathbb{R}$ is given by

$$\Phi(y) = \|y\|_2^2.$$

Observe that if y is a probability vector, then $\Phi(y)$ is a measure of how far y is from the uniform distribution. In this respect, $\Phi(\cdot)$ is similar to the potential function Φ_i discussed in Section 3.4.1; however, compared to Φ_i, our function $\Phi(\cdot)$ is more compatible with spectral analysis.

As discussed previously, we wish to argue that in each step, we fall into one of two cases: the error-free case, and the mixing case. To implement this plan, let us think of $|\Phi(z^{[j+2]}) - \Phi(\widetilde{z}^{[j+2]})|$ as the *amount of error in step j*. (Intuitively, if we pretend that $z^{[j]}$ is a probability vector, then the quantity $|\Phi(z^{[j+2]}) - \Phi(\widetilde{z}^{[j+2]})|$ measures the extent to which $\Delta \mathbf{W}^{(1)}_{j+2\leftarrow j}$ "damages mixed-ness," since $\Phi(\cdot)$ is our measure of mixing.) Using the fact that $\widetilde{\mathbf{W}}_{j+2\leftarrow j} \overset{\text{SV}}{\approx}_{\tau_0} \mathbf{W}_{j+2\leftarrow j}$, we prove that

$$\underbrace{\left|\Phi(z^{[j+2]}) - \Phi(\widetilde{z}^{[j+2]})\right|}_{\text{amount of error in step } j} \le O(\tau_0) \cdot \left(\underbrace{\Phi(z^{[j]}) - \Phi(\mathbf{W}_{j+2\leftarrow j} \cdot z^{[j]})}_{\text{mixing in } z} + \underbrace{\Phi(x^{[j]}) - \Phi(\mathbf{W}_{j+2\leftarrow j} \cdot x^{[j]})}_{\text{mixing in } x} \right). \quad (7)$$

Thus, if the amount of error $|\Phi(z^{[j+2]}) - \Phi(\widetilde{z}^{[j+2]})|$ is large, then there must be a corresponding change in the potential function: either $\Phi(\mathbf{W}_{j+2\leftarrow j} \cdot z^{[j]}) \ll \Phi(z^{[j]})$, or else $\Phi(\mathbf{W}_{j+2\leftarrow j} \cdot x^{[j]}) \ll \Phi(x^{[j]})$. The first case describes mixing that occurs "on the z side," i.e., after multiplying by $\mathbf{L}^{-1}\Delta \mathbf{W}^{(1)}$. The second case describes mixing that occurs "on the x side," i.e., the mixing was already present in the test vector x to begin with. Either way, we see that *error at step j implies mixing at step j.*

The $\mathbf{F}_1$-seminorm. In light of (7), it is natural to define our "norm function" $\|\cdot\| : \mathbb{R}^{(n+1)\cdot w} \to \mathbb{R}$ by the formula

$$\|y\|^2 = \sum_{j \in \{0,2,4,\ldots,n\}} \left(\Phi(y^{[j]}) - \Phi(\mathbf{W}_{j+2 \leftarrow j} \cdot y^{[j]}) \right),$$

where $y = (y^{[0]}, \ldots, y^{[n]})$. This quantity $\|y\|^2$ measures the *total potential drops* when we apply each matrix $\mathbf{W}_{j+2\leftarrow j}$ to the appropriate block of y. One can verify that this function $\|\cdot\|$ truly is a (semi)norm; we refer to it as the "$\mathbf{F}_1$-seminorm." Summing up (7) gives

$$\sum_{j \in \{0,2,4,\ldots,n\}} \underbrace{\left| \Phi(z^{[j+2]}) - \Phi(\widetilde{z}^{[j+2]}) \right|}_{\text{amount of error in step } j} \leq O(\tau) \cdot (\|z\|^2 + \|x\|^2). \tag{8}$$

The left-hand side above is a measure of the total amount of error. However, our job is to bound $\|z\|$, which is, conceptually, a different way of measuring the total amount of error. To connect these two quantities, it might be helpful to visualize a dynamic process generating $z^{[0]}, \widetilde{z}^{[2]}, z^{[2]}, \ldots$ as in Figure 3.

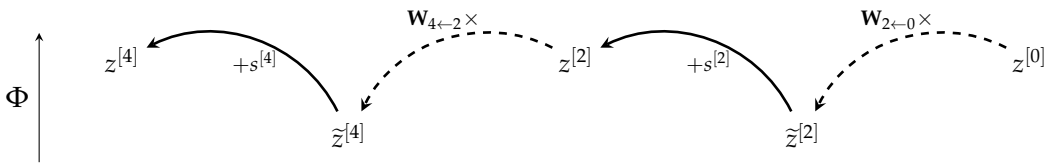

Figure 3: An illustration of the dynamics of the potential function.

The value $\|z\|^2$ is the sum of the heights of the dashed arrows in Figure 3. Meanwhile, the error measure on the left-hand side of (8) is the sum of the heights of the solid arrows in Figure 3. The key observation is that in this process, *the initial point $z^{[0]}$ is equal to the final point $z^{[n+2]}$* (both are the zero vector), so the two types of arrows must perfectly balance. Symbolically, we have

$$\begin{aligned}
\|z\|^2 &= \sum_{j \in \{0,2,\ldots,n\}} \left(\underbrace{\Phi(z^{[j]}) - \Phi(\widetilde{z}^{[j+2]})}_{\text{drop in potential}} \right) \\
&= \Phi(z^{[0]}) - \Phi(z^{[n+2]}) + \sum_{j \in \{0,2,\ldots,n\}} \left(\Phi(z^{[j+2]}) - \Phi(\widetilde{z}^{[j+2]}) \right) \\
&\leq \Phi(z^{[0]}) - \Phi(z^{[n+2]}) + \sum_{j \in \{0,2,\ldots,n\}} \underbrace{\left| \Phi(z^{[j+2]}) - \Phi(\widetilde{z}^{[j+2]}) \right|}_{\text{amount of error in step } j} \\
&= 0 - 0 + O(\tau) \cdot (\|z\|^2 + \|x\|^2)
\end{aligned}$$

by (8). By choosing τ to be sufficiently small, we conclude that $\|z\| \leq O(\sqrt{\tau} \cdot \|x\|)$ as desired.

In the actual proof, to account for $\Delta\mathbf{W}^{(t)}$ where $t > 1$, we use a somewhat more complicated seminorm called the "$\mathbf{F}$-seminorm." Intuitively, the $\mathbf{F}$-seminorm measures the total potential drops across all edges $(i, j) \in E(\mathsf{SC}_n)$, whereas the $\mathbf{F}_1$-seminorm only looks at edges of length 2. See Section 8 for details.

3.5 Our WPRG for Unbounded-Width Permutation ROBPs

We conclude this technical overview by briefly discussing our improved WPRG for unbounded-width permutation ROBPs (Theorem 1.9). This WPRG corresponds closely to our simplified proof of Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's main theorem [AKMPSV20]. We construct our WPRG by reverse-engineering the (non-black-box) algorithm for derandomizing regular ROBPs. In order to carry out this reverse-engineering process, we assume that the program we are fooling is a *permutation* program, because under this assumption, Rozenman and Vadhan's derandomized squaring operation [RV05] has a "black-box interpretation," namely, it is equivalent to the Impagliazzo-Nisan-Wigderson (INW) PRG [INW94]. The WPRG that we construct in this way has a seed length that is greater than what we can afford, because each seed includes several independent seeds for the INW PRG. We therefore decrease the seed length by using Hoza, Pyne, and Vadhan's PRG [HPV21] to generate correlated seeds for the INW PRG.

The approach outlined above is essentially identical to the approach that Pyne and Vadhan use to construct their WPRG for permutation ROBPs [PV21]. The reason we get an improved bottom-line seed length is that Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's algorithm [AKMPSV20] involves a "cycle lift" at some points. These cycle lifts are inherited by Pyne and Vadhan [PV21], and they effectively cost a factor of n in the error parameter. In terms of seed length, this means that $O(\log n \cdot \sqrt{\log(1/\varepsilon)})$ becomes $O(\log^{3/2} n + \log n \cdot \sqrt{\log(1/\varepsilon)})$. In contrast, we never use any cycle lifts, hence we do not pay the extra $\log^{3/2} n$ term.

4 Our Error Reduction Framework: Inverse Analysis of Random Walks and Shortcutting

In this section, we rigorously describe our general error reduction framework for space-bounded derandomization. All of our results rely on this framework.

4.1 When Our Framework Is Applicable

We follow the same notation as in Section 3. Let B be a width-w length-n standard-order ROBP. The vertex set of B is $V(B) := V^{(0)} \cup V^{(1)} \cup \cdots \cup V^{(n)}$ where $|V^{(i)}| = w$ for each $i \in [n]$. The matrices $\mathbf{W}$ and $\mathbf{L} := \mathbf{I}_{(n+1)w} - \mathbf{W}$ are also defined as in Section 3.

We are interested in deterministically approximating the matrix $\mathbf{L}^{-1}$. Our error-reduction framework is a method for converting an appropriate "moderate-error" approximation into a "low-error" approximation. We begin by explaining exactly what type of "moderate-error" initial approximation our framework requires.

4.1.1 Shortcutting and Approximation Ensembles

We assume that we start with an *ensemble* of moderate-error approximation matrices – one for each edge in the "shortcut graph," which we introduced in Section 3 (see Figure 1) and which we formally define next.

Definition 4.1 (Shortcut Graph)**.** *Let $n \geq 1$ be a power of two. The shortcut graph SC_n (or SC when n is clear from the context) of size n is a directed graph with $n+1$ vertices $V(\mathsf{SC}_n) = \{0, 1, \ldots, n\}$ and the following set of edges. For each $t \in \{0, 1, \ldots, \log n\}$, denote*

$$U_t = \{j \cdot 2^t : j \in \mathbb{Z} \cap [0, n/2^t]\} = \{0, 2^t, 2 \cdot 2^t, 3 \cdot 2^t, \ldots, n\}.$$

Then, $E(\mathsf{SC}_n)$ contains one directed edge $(j, j+2^t)$ for every adjacent pair $\{j, j+2^t\} \subseteq U_t$. Note that $|E(\mathsf{SC}_n)| = 2n-1$.

Recall that $\mathbf{W}_{j\leftarrow i} = \mathbf{W}_j \cdot \mathbf{W}_{j-1} \cdots \mathbf{W}_{i+1}$, so each entry of $\mathbf{W}_{j\leftarrow i}$ describes the probability that a length-$(j-i)$ random walk from a particular vertex in $V^{(i)}$ arrives at a particular vertex in $V^{(j)}$. To use our error reduction framework, we must start with a moderate-error approximation matrix $\widetilde{\mathbf{W}}_{j\leftarrow i}$ for each edge $(i,j) \in E(\mathsf{SC}_n)$.

Definition 4.2 (Approximation Ensemble)**.** *A (w,n) approximation ensemble is a collection of matrices $\widetilde{\mathcal{W}} = \{\widetilde{\mathbf{W}}_{j\leftarrow i} : (i,j) \in E(\mathsf{SC}_n)\}$ where $\widetilde{\mathbf{W}}_{j\leftarrow i} \in \mathbb{R}^{w\times w}$. We omit the parameters (w,n) when they are clear from context.*

Looking ahead, we will obtain this ensemble in different ways depending on our goal.

- When constructing WPRGs, for each $(i,j) \in E(\mathsf{SC})$, we evaluate a given PRG $\mathcal{G} : \{0,1\}^s \to \{0,1\}^n$ on the subprogram from the i-th layer to the j-th layer. As the subprogram only reads $(j-i)$ bits, we only need to use the first $j-i$ pseudorandom output bits. The details are given in Section 5.
- For estimating random walks (i.e., non-black-box derandomization), we will use the derandomized square operation by Rozenman and Vadhan [RV05]. The details can be found in Appendix A.

4.1.2 The Correction Graph and the Appropriate Initial Error Bound

Our error reduction framework assumes that the initial approximation ensemble $\widetilde{\mathcal{W}}$ already has a "moderately small error." The specific "error" measure is defined in terms of the *correction graph*, defined next.

Definition 4.3 (Correction Graph)**.** *Let $\widetilde{\mathcal{W}} = \{\widetilde{\mathbf{W}}_{j\leftarrow i} : (i,j) \in E(\mathsf{SC}_n)\}$ be an approximation ensemble. To define the corresponding* correction graph transition matrix *$\Delta\mathbf{W} \in (\mathbb{R}^{w\times w})^{(n+1)\times(n+1)}$, let $\ell, r \in \{0,\ldots,n\}$. The (r,ℓ)-th block of $\Delta\mathbf{W}$ is defined as follows.*

- *If $(\ell, r) \in E(\mathsf{SC}_n)$, then*

$$(\Delta\mathbf{W})_{r\leftarrow\ell} = \begin{cases} \widetilde{\mathbf{W}}_{r\leftarrow\ell} - \mathbf{W}_{r\leftarrow\ell} & \text{if } r = \ell+1, \\ \widetilde{\mathbf{W}}_{r\leftarrow\ell} - \widetilde{\mathbf{W}}_{r\leftarrow\frac{\ell+r}{2}}\widetilde{\mathbf{W}}_{\frac{\ell+r}{2}\leftarrow\ell} & \text{otherwise.} \end{cases}$$

- *If $(\ell, r) \notin E(\mathsf{SC}_n)$, then $(\Delta\mathbf{W})_{r\leftarrow\ell} = \mathbf{0}$.*

The correction graph *is a weighted digraph on the vertex set $V(B)$ with the edge weights described by $\Delta\mathbf{W}$.*

Our error measure for $\widetilde{\mathcal{W}}$ is $\|\mathbf{L}^{-1}\Delta\mathbf{W}\|$, where $\|\cdot\|$ is an arbitrary submultiplicative matrix norm, or more generally, an arbitrary extended submultiplicative matrix seminorm (see Definition 2.3). Our error reduction procedure is applicable provided that $\|\mathbf{L}^{-1}\Delta\mathbf{W}\|$ is moderately small. Under this assumption, we will show how to construct a matrix $\mathbf{A}_m$ such that $\|\mathbf{I} - \mathbf{A}_m\mathbf{L}\|$ is extremely small. In this sense, $\mathbf{A}_m$ will be a low-error approximation to $\mathbf{L}^{-1}$.

Looking ahead, to establish the initial moderate bound on $\|\mathbf{L}^{-1}\Delta\mathbf{W}\|$, we will frequently decompose $\Delta\mathbf{W}$ into

$$\Delta\mathbf{W} = \sum_{t=0}^{\log n} \Delta\mathbf{W}^{(t)} \tag{9}$$

where for each t, we define $\Delta\mathbf{W}^{(t)}$ as

$$(\Delta\mathbf{W})^{(t)}_{r\leftarrow\ell} = \begin{cases} (\Delta\mathbf{W})_{r\leftarrow\ell} & \text{if } r = \ell + 2^t \\ 0 & \text{otherwise.} \end{cases}$$

That is, $\Delta\mathbf{W}^{(t)}$ contains all those blocks that correspond to "level-t" edges in SC_n. Then, we will reason about each term $\|\mathbf{L}^{-1}\Delta\mathbf{W}^{(t)}\|$ separately. Furthermore, for convenience, we will usually ensure that $\widetilde{\mathbf{W}}_{\ell+1\leftarrow\ell} = \mathbf{W}_{\ell+1\leftarrow\ell}$ for each $\ell \in \{0, 1, \ldots, n-1\}$, so $(\Delta\mathbf{W})^{(0)} = \mathbf{0}$.

4.2 The Error Reduction Construction

4.2.1 Moderate-Error Inverse Laplacian $\widehat{\mathbf{L}}^{-1}$: Shortcutting

Let $\widetilde{\mathcal{W}}$ be an approximation ensemble. The first step of the error reduction process is to construct a matrix $\widehat{\mathbf{L}}^{-1}$, intended to be a moderate-error approximate inverse to the Laplacian matrix $\mathbf{L}$. The construction is as follows. We view $\widehat{\mathbf{L}}^{-1}$ as block matrix from $(\mathbb{R}^{w\times w})^{(n+1)\times(n+1)}$, and we use $(\widehat{\mathbf{L}}^{-1})_{j\leftarrow i}$ to denote the (j,i) block (a $w \times w$ matrix).

For each $\ell < r$, let $\ell = i_0 \to i_1 \to \cdots \to i_k = r$ be the shortest path from ℓ to r in SC. (There is indeed a *unique* shortest path; see Corollary 4.8.) We construct the (r,ℓ)-th block of $\widehat{\mathbf{L}}^{-1}$ as

$$(\widehat{\mathbf{L}}^{-1})_{r\leftarrow\ell} \coloneqq \widetilde{\mathbf{W}}_{i_k\leftarrow i_{k-1}} \cdot \widetilde{\mathbf{W}}_{i_{k-1}\leftarrow i_{k-2}} \cdots \widetilde{\mathbf{W}}_{i_2\leftarrow i_1} \cdot \widetilde{\mathbf{W}}_{i_1\leftarrow i_0}. \tag{10}$$

We also set $(\widehat{\mathbf{L}}^{-1})_{\ell\leftarrow\ell} = \mathbf{I}_w$ for every $\ell \in [0,n]$, and $(\widehat{\mathbf{L}}^{-1})_{r\leftarrow\ell} = \mathbf{0}$ for every $\ell > r$. This completes the construction of $\widehat{\mathbf{L}}^{-1}$.

4.2.2 Low-Error Inverse Laplacian $\mathbf{A}_m$: Richardson Iteration

The final step of our error reduction framework is that we apply Richardson iteration to $\widehat{\mathbf{L}}^{-1}$. That is, for each $m \in \mathbb{N}$, define

$$\mathbf{A}_m = \sum_{i=0}^{m} (\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L})^i \widehat{\mathbf{L}}^{-1}.$$

The main theorem of this section says that $\mathbf{A}_m$ is a low-error approximate inverse to the Laplacian matrix $\mathbf{L}$:

Theorem 4.4 (General Error Reduction Framework)**.** *Let B be a width-w length-n standard-order ROBP with transition matrix $\mathbf{W}$ and Laplacian matrix $\mathbf{L} = \mathbf{I} - \mathbf{W}$. Let $\widetilde{\mathcal{W}} = \{\widetilde{\mathbf{W}}_{j\leftarrow i} : (i,j) \in \mathsf{SC}_n\}$ be an approximation ensemble, let $m \in \mathbb{N}$, and let $\Delta\mathbf{W}$ and $\mathbf{A}_m$ be the corresponding matrices defined above. Let $\|\cdot\|$ be an extended submultiplicative matrix seminorm, and let $0 < \delta \le 1/2$. If $\|\mathbf{L}^{-1}\Delta\mathbf{W}\| \le \delta$, then $\|\mathbf{I} - \mathbf{A}_m \cdot \mathbf{L}\| \le (2\delta)^{m+1}$.*

The rest of Section 4 is devoted to proving Theorem 4.4.

4.3 Correction Graph Lemma: Inverse Analysis of Random Walks

Since $\widehat{\mathbf{L}}^{-1}$ is a unitriangular matrix, it is invertible. We define $\widehat{\mathbf{L}}$ as the inverse of $\widehat{\mathbf{L}}^{-1}$. The core of the proof of Theorem 4.4 is the following lemma, which gives an exact formula for $\widehat{\mathbf{L}}$. This process of going from $\widehat{\mathbf{L}}^{-1}$ to $\widehat{\mathbf{L}}$ is what we refer to as the "inverse analysis of random walks."

Lemma 4.5 (Correction Graph Lemma)**.** *Let B be a width-w length-n standard-order ROBP with transition matrix $\mathbf{W}$ and Laplacian matrix $\mathbf{L} = \mathbf{I} - \mathbf{W}$. Let $\widetilde{\mathcal{W}} = \{\widetilde{\mathbf{W}}_{j\leftarrow i} : (i,j) \in \mathsf{SC}_n\}$ be an approximation ensemble, and let $\Delta\mathbf{W}$ and $\widehat{\mathbf{L}}$ be the corresponding matrices defined above. Define $\widehat{\mathbf{W}} = \mathbf{W} + \Delta\mathbf{W}$. Then*

$$\widehat{\mathbf{L}} = \mathbf{I} - \widehat{\mathbf{W}}.$$

Intuitively, Lemma 4.5 says that $\widehat{\mathbf{L}}$ can be interpreted as the "Laplacian" of a weighted graph with transition matrix $\widehat{\mathbf{W}} = \mathbf{W} + \Delta\mathbf{W}$. This is the justification for the term "correction graph:" by subtracting the correction graph from $\widehat{\mathbf{L}}$, we get the original Laplacian $\mathbf{L}$.

The first step in the proof of Lemma 4.5 is to give a formula for each block of $(\mathbf{I} - \widehat{\mathbf{W}})^{-1}$. Intuitively, because $\mathbf{I} - \widehat{\mathbf{W}}$ is a directed Laplacian matrix, a block $(\mathbf{I} - \widehat{\mathbf{W}})^{-1}_{r\leftarrow\ell}$ should describe random walks starting from layer ℓ and ending at layer r. This "random walk" interpretation is slightly inaccurate because $\widehat{\mathbf{W}}$ is not a stochastic matrix, but otherwise it is correct:

Claim 4.6. *The matrix $\mathbf{I} - \widehat{\mathbf{W}}$ is invertible, and for each $\ell, r \in \{0,\dots,n\}$, the (ℓ,r)-th block of $(\mathbf{I} - \widehat{\mathbf{W}})^{-1}$ is given by*

$$(\mathbf{I} - \widehat{\mathbf{W}})^{-1}_{r\leftarrow\ell} = \sum_{t=0}^{r-\ell} \sum_{\substack{\ell = i_0 < i_1 < \cdots < i_t = r \\ (i_0,\dots,i_t) \text{ is a path in } \mathsf{SC}}} \widehat{\mathbf{W}}_{i_t\leftarrow i_{t-1}} \cdot \widehat{\mathbf{W}}_{i_{t-1}\leftarrow i_{t-2}} \cdots \widehat{\mathbf{W}}_{i_1\leftarrow i_0}. \tag{11}$$

Proof. Since $\mathbf{I} - \widehat{\mathbf{W}}$ is unitriangular, it is indeed invertible, and its inverse is given by

$$(\mathbf{I} - \widehat{\mathbf{W}})^{-1} = \sum_{t=0}^{\infty} \widehat{\mathbf{W}}^t.$$

Therefore, the (ℓ,r)-th block of $(\mathbf{I} - \widehat{\mathbf{W}})^{-1}$ is given by

$$(\mathbf{I} - \widehat{\mathbf{W}})^{-1}_{r\leftarrow\ell} = \sum_{t=0}^{\infty} (\widehat{\mathbf{W}}^t)_{r\leftarrow\ell} = \sum_{t=0}^{\infty} \sum_{\ell = i_0, i_1, \dots, i_t = r} \widehat{\mathbf{W}}_{i_t\leftarrow i_{t-1}} \cdot \widehat{\mathbf{W}}_{i_{t-1}\leftarrow i_{t-2}} \cdots \widehat{\mathbf{W}}_{i_1\leftarrow i_0}.$$

The block $\widehat{\mathbf{W}}_{j\leftarrow i}$ is zero whenever $(i,j) \notin E(\mathsf{SC}_n)$, and hence the sum above is equivalent to (11). □

We would like to simplify the formula in (11). We will do so using the following two structural properties of the shortcut graph SC_n.

Lemma 4.7 (Intermediate Value Lemma)**.** *Suppose $(i,j) \in E(\mathsf{SC})$. Further, suppose a path π in SC starts at i and visits two vertices a, b where $i < a < j \le b$. Then, the path also visits j.*

Proof. Without loss of generality, we may assume that $(a,b) \in E(\mathsf{SC})$. Under this assumption, we will show that $b = j$. By the fact that (a,b) is an edge, we know that a, b are two adjacent multiples of 2^q for some $q \in [\log n]$. Similarly, by the fact that (i,j) is an edge, we know that i, j are two adjacent multiples of $2^{q'}$ for some $q' \in [\log n]$. Since $i < a < j$, the value a cannot be a multiple of $2^{q'}$, and since a is a multiple of 2^q, we get $q < q'$. But then both j and a are multiples of 2^q. Since $a < b$ are two adjacent multiplies of 2^q, it must be the case that $b \le j$. So we conclude $b = j$. □

As a corollary of the Intermediate Value Lemma, we have the following characterization of shortest paths in SC.

Corollary 4.8 (Characterization of shortest paths)**.** *Let $\ell, r, m \in \{0,1,\dots,n\}$. Suppose the shortest path from ℓ to r in SC visits m. Then, every path from ℓ to r in SC visits m.*

Proof. Let π and π' be two paths from ℓ to r, and suppose that π visits m but π' does not visit m. We will show that π is not the shortest path from ℓ to r.

Let f be the *first* vertex that π visits but π' does not visit. Let a be the preceding vertex in π, i.e., π traverses an edge $(a, f) \in E(\mathsf{SC})$. By the Intermediate Value Lemma on the edge (a, f), π' goes from a directly to some $b > f$ (otherwise it could not avoid visiting f). Then, we apply the Intermediate Value Lemma again on the edge (a, b). We know π must visit b at some point. Therefore, π cannot be the shortest path, because we could go from a directly to b instead of taking a pit stop at f. □

Observe that Corollary 4.8 also implies that the shortest path from ℓ to r is unique.

Now we are ready to simplify (11). In (11), we sum over all paths from ℓ to r in SC. We now give an equivalent formula that only considers the *shortest* path from ℓ to r.

Claim 4.9. *Let $0 \le \ell < r \le n$, and let $\ell = i_0 \to i_1 \to \cdots \to i_t = r$ be the shortest path from ℓ to r in SC. The (ℓ, r)-th block of $(\mathbf{I} - \widehat{\mathbf{W}})^{-1}$ is given by*

$$(\mathbf{I} - \widehat{\mathbf{W}})^{-1}_{r \leftarrow \ell} = \widetilde{\mathbf{W}}_{i_t \leftarrow i_{t-1}} \cdot \widetilde{\mathbf{W}}_{i_{t-1}} \cdots \widetilde{\mathbf{W}}_{i_1 \leftarrow i_0}. \tag{12}$$

Proof. First, suppose that $t = 1$, i.e., $(\ell, r) \in E(\mathsf{SC}_n)$. Then the difference $|r - \ell|$ must be a power of two, say $r = \ell + 2^q$. We proceed by induction on q. If $q = 0$, then there is exactly one path from ℓ to r in SC, so the right-hand-side of (11) simplifies to $\widehat{\mathbf{W}}_{r \leftarrow \ell} = \widetilde{\mathbf{W}}_{r \leftarrow \ell}$, completing the proof in this case. Suppose now that $q > 0$.

Let $m = \ell + 2^{q-1}$ (the midpoint). Every path from ℓ to r of length greater than 1 must visit m. (Indeed, either the first step is from ℓ to m, or else we can apply the Intermediate Value Lemma.) Therefore, by Claim 4.6, we have

$$\begin{aligned}
(\mathbf{I} - \widehat{\mathbf{W}})^{-1}_{r \leftarrow \ell} &= \widehat{\mathbf{W}}_{r \leftarrow \ell} + (\mathbf{I} - \widehat{\mathbf{W}})^{-1}_{r \leftarrow m} \cdot (\mathbf{I} - \widehat{\mathbf{W}})^{-1}_{m \leftarrow \ell} \\
&= \widehat{\mathbf{W}}_{r \leftarrow \ell} + \widetilde{\mathbf{W}}_{r \leftarrow m} \cdot \widetilde{\mathbf{W}}_{m \leftarrow \ell} && \text{(Induction)} \\
&= \widetilde{\mathbf{W}}_{r \leftarrow \ell} && \text{(By the definition of } \widehat{\mathbf{W}}.)
\end{aligned}$$

That completes the proof of the case $t = 1$, which is the base case of induction on t. For the inductive step, suppose $t > 1$. Note that the shortest path from ℓ to i_1 is a single step, and the shortest path from i_1 to r is $i_1 \to i_2 \to \cdots \to i_{k-1} \to r$. By Corollary 4.8, *every* path from ℓ to r visits i_1, so by Claim 4.6,

$$\begin{aligned}
(\mathbf{I} - \widehat{\mathbf{W}})^{-1}_{r \leftarrow \ell} &= (\mathbf{I} - \widehat{\mathbf{W}})^{-1}_{r \leftarrow i_1} \cdot (\mathbf{I} - \widehat{\mathbf{W}})^{-1}_{i_1 \leftarrow \ell} \\
&= (\widetilde{\mathbf{W}}_{r \leftarrow i_{k-1}} \cdots \widetilde{\mathbf{W}}_{i_2 \leftarrow i_1}) \cdot \widetilde{\mathbf{W}}_{i_1 \leftarrow \ell}
\end{aligned}$$

by induction. □

Proof of Lemma 4.5. Claim 4.9 shows that when $\ell < r$, the (ℓ, r)-th block of $(\mathbf{I} - \widehat{\mathbf{W}})^{-1}$ is equal to the corresponding block of $\widehat{\mathbf{L}}^{-1}$. Looking at Claim 4.6, we see that the same holds when $\ell < r$ (the right-hand side of (11) is an empty sum, i.e., the zero matrix) and when $\ell = r$ (the right-hand side of (11) is an empty product, i.e., the identity matrix). Therefore, $(\mathbf{I} - \widehat{\mathbf{W}})^{-1} = \widehat{\mathbf{L}}^{-1}$. Since matrix inverses are unique, it follows that $\widehat{\mathbf{L}} = \mathbf{I} - \widehat{\mathbf{W}}$. □

4.4 Using the Correction Graph Lemma to Complete the Error Reduction Proof

Recall that we are working on proving our general error reduction theorem (Theorem 4.4). The assumption in Theorem 4.4 says that $\|\mathbf{L}^{-1}\Delta\mathbf{W}\|$ is moderately small. The Correction Graph Lemma (Lemma 4.5) says that $\Delta\mathbf{W} = \Delta\mathbf{L} := \mathbf{L} - \widehat{\mathbf{L}}$, so now we know that $\|\mathbf{L}^{-1}\Delta\mathbf{L}\|$ is moderately small. Our next goal is to show that it follows that $\|\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L}\|$ is moderately small. This is a consequence of the following natural lemma.

Lemma 4.10 (Inverse of approximate identity also approximates identity)**.** *Let $\|\cdot\|$ be an extended submultiplicative matrix seminorm, let $\mathbf{R} \in \mathbb{R}^{N\times N}$ be a unitriangular matrix, and let $\delta \in [0,1)$ be a real number. If $\|\mathbf{R} - \mathbf{I}\| \le \delta$, then $\|\mathbf{R}^{-1} - \mathbf{I}\| \le \delta/(1-\delta)$.*

Proof. Let $\mathbf{E} = \mathbf{I} - \mathbf{R}$. Since $\mathbf{R}$ is unitriangular, $\mathbf{E}^N = \mathbf{0}$. Therefore, if we let $\mathbf{S} = \sum_{i=1}^{N-1}\mathbf{E}^i$, then $\mathbf{E}\cdot\mathbf{S} = \mathbf{S} - \mathbf{E}$. Rearranging, we get

$$\mathbf{S} = (\mathbf{I} - \mathbf{E})^{-1}\cdot\mathbf{E} = \mathbf{R}^{-1}\cdot(\mathbf{I} - \mathbf{R}) = \mathbf{R}^{-1} - \mathbf{I}.$$

Therefore,

$$\|\mathbf{R}^{-1} - \mathbf{I}\| = \|\mathbf{S}\| \le \sum_{i=1}^{N-1}\delta^i \le \sum_{i=1}^{\infty}\delta^i = \frac{\delta}{1-\delta}. \qquad \square$$

Corollary 4.11. *Let $\mathbf{L}$ and $\widehat{\mathbf{L}}$ be lower unitriangular matrices of the same dimension, let $\delta \in [0,1)$, and let $\|\cdot\|$ be an extended submultiplicative matrix seminorm. Suppose $\|\mathbf{L}^{-1}(\widehat{\mathbf{L}} - \mathbf{L})\| \le \delta$. Then $\|\widehat{\mathbf{L}}^{-1}(\widehat{\mathbf{L}} - \mathbf{L})\| \le \delta/(1-\delta)$.*

Proof. Take $\mathbf{R} = \mathbf{L}^{-1}\cdot\widehat{\mathbf{L}}$ in the previous lemma. This works, because $\mathbf{R}$ is unitriangular, $\mathbf{L}^{-1}(\widehat{\mathbf{L}} - \mathbf{L}) = \mathbf{R} - \mathbf{I}$, and $\widehat{\mathbf{L}}^{-1}(\widehat{\mathbf{L}} - \mathbf{L}) = \mathbf{I} - \mathbf{R}^{-1}$. $\square$

Note that $\widehat{\mathbf{L}}^{-1}(\widehat{\mathbf{L}} - \mathbf{L}) = \mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L}$. Given that $\|\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L}\|$ is moderately small, the standard analysis of Richardson iteration shows that $\|\mathbf{I} - \mathbf{A}_m\mathbf{L}\|$ is very small:

Lemma 4.12 (Richardson Iteration)**.** *Let $\mathbf{L}$ and $\widehat{\mathbf{L}}^{-1}$ be square matrices of the same dimension and let $\mathbf{A}_m = \sum_{i=0}^{m}(\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L})^i\widehat{\mathbf{L}}^{-1}$. Let $\|\cdot\|$ be an extended submultiplicative matrix seminorm, and assume $\|\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L}\| < \infty$. Then*

$$\|\mathbf{I} - \mathbf{A}_m\mathbf{L}\| \le \|\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L}\|^{m+1}.$$

Proof.

$$\begin{aligned}
\mathbf{I} - \mathbf{A}_m\mathbf{L} &= \mathbf{I} - \sum_{i=0}^{m}(\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L})^i\cdot\widehat{\mathbf{L}}^{-1}\mathbf{L} \\
&= (\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L}) - \sum_{i=1}^{m}(\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L})^i\cdot\widehat{\mathbf{L}}^{-1}\mathbf{L} && \text{(Taking out the } i=0 \text{ term)} \\
&= (\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L})^2 - \sum_{i=2}^{m}(\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L})^i\cdot\widehat{\mathbf{L}}^{-1}\mathbf{L} && \text{(Taking out the } i=1 \text{ term)} \\
&\vdots \\
&= (\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L})^{m+1}.
\end{aligned}$$

$\square$

Proof of Theorem 4.4. By the Correction Graph Lemma (Lemma 4.5), we have $\|\mathbf{L}^{-1}(\widehat{\mathbf{L}} - \mathbf{L})\| \le \delta$. The matrices $\mathbf{L}$ and $\widehat{\mathbf{L}}$ are both lower unitriangular. By Corollary 4.11, it follows that $\|\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L}\| \le \delta/(1-\delta) \le 2\delta$. By Lemma 4.12, it follows that $\|\mathbf{I} - \mathbf{A}_m\mathbf{L}\| \le (2\delta)^{m+1}$. $\square$

5 The Black-Box Version of Our Error Reduction Framework: Low-Error WPRGs

In the previous section, we described our error reduction framework in its most general form: given a moderate-error initial approximation ensemble $\widetilde{\mathcal{W}}$, we showed how to construct a low-error approximation matrix $\mathbf{A}_m$. In this section, we "translate" our error reduction framework into the black-box setting. That is, we consider the case that $\widetilde{\mathcal{W}}$ is induced by a moderate-error PRG $\mathcal{G}$. In this case, we will show that the low-error approximation matrix $\mathbf{A}_m$ can be "implemented" by a WPRG. Thus, we can convert a moderate-error PRG into a low-error WPRG.

Looking ahead, the error reduction procedure that we describe in this section will be the *first of two stages* in our constructions of low-error WPRGs for bounded-width regular ROBPs (Section 6) and for unbounded-width permutation ROBPs (Section 9). The WPRG that we construct in this first stage will have low error, but it will also have a relatively large seed length, because the seed of the WPRG will include (among other things) several independent seeds of $\mathcal{G}$. In the second stage, we will decrease the seed length by using the INW PRG [INW94] to select correlated seeds of $\mathcal{G}$. This second stage is the same as in prior work [CDRST21; PV21]; the novelty of our construction is how we perform the first stage, where we use the machinery developed in the previous section.

5.1 The Low-Error WPRG Construction

To describe the WPRG construction formally, we will work with the pseudodistribution formalism described by Hoza [Hoz21].

Definition 5.1 (Pseudodistributions)**.** *Let $n \in \mathbb{N}$. A* pseudodistribution *over $\{0,1\}^n$ is a formal real linear combination $\mathcal{X}$ of n-bit strings, i.e., $\mathcal{X} = \sum_{i=1}^{R} \lambda_i \cdot x^{(i)}$ where $\lambda_i \in \mathbb{R}$ and $x^{(i)} \in \{0,1\}^n$. If every λ_i is nonnegative and $\sum_{i=1}^{R} \lambda_i = 1$, then $\mathcal{X}$ can be interpreted as a true probability distribution. For example,*

$$\mathcal{U}_n = \sum_{x \in \{0,1\}^n} 2^{-n} \cdot x.$$

A linear combination of pseudodistributions is another pseudodistribution defined in the natural way. That is, if $\mathcal{X}$ and $\mathcal{Y}$ are both pseudodistributions over $\{0,1\}^n$, say $\mathcal{X} = \sum_{i=1}^{R} \lambda_i \cdot x^{(i)}$ and $\mathcal{Y} = \sum_{i=1}^{K} \nu_i \cdot y^{(i)}$, and $c \in \mathbb{R}$, then $\mathcal{X} + c\mathcal{Y} = \sum_{i=1}^{R+K} \eta_i \cdot z^{(i)}$, where

$$\begin{aligned} &\eta_i = \lambda_i \quad \text{and} \quad z^{(i)} = x^{(i)} && \text{if } i \leq R \\ &\eta_i = c \cdot \nu_{i-R} \quad \text{and} \quad z^{(i)} = y^{(i-R)} && \text{if } i > R. \end{aligned}$$

The tensor product *of two pseudodistributions is defined by*

$$\left(\sum_{i=1}^{R} \lambda_i \cdot x^{(i)} \right) \otimes \left(\sum_{j=1}^{K} \nu_j \cdot y^{(j)} \right) = \sum_{i=1}^{R} \sum_{j=1}^{K} (\lambda_i \nu_j) \cdot (x^{(i)} \circ y^{(j)}),$$

where $x \circ y$ denotes the concatenation of x with y. Thus, if $\mathcal{X}$ is a pseudodistribution over $\{0,1\}^{n_1}$ and $\mathcal{Y}$ is a pseudodistribution over $\{0,1\}^{n_2}$, then the tensor product $\mathcal{X} \otimes \mathcal{Y}$ is a pseudodistribution over $\{0,1\}^{n_1+n_2}$.

A pseudodistribution with R terms corresponds to a WPRG with seed length $\log R$.

Converting a moderate-error PRG into a low-error pseudodistribution. Fix $\mathcal{G}\colon \{0,1\}^s \to \{0,1\}^n$, which we think of as an initial "moderate-error" PRG. We identify the PRG $\mathcal{G} : \{0,1\}^s \to \{0,1\}^n$ with the pseudorandom distribution that it samples, viewed as a pseudodistribution over $\{0,1\}^n$:

$$\mathcal{G} = \sum_{x \in \{0,1\}^s} 2^{-s} \cdot \mathcal{G}(x).$$

For each $1 \le i \le j \le n$, let $\mathcal{G}_{i \to j}$ denote[11] the pseudodistribution over $\{0,1\}^d$ obtained by taking the $(j-i)$-bit prefix of a sample from $\mathcal{G}$, i.e.,

$$\mathcal{G}_{i \to j} = \sum_{x \in \{0,1\}^s} 2^{-s} \cdot (\mathcal{G}(x)_1, \ldots, \mathcal{G}(x)_{j-i}).$$

Furthermore, for any indices $0 \le i < j \le n$, let $i = i_0 \to i_1 \to \cdots \to i_t = j$ be the shortest path from i to j through SC, and define

$$\mathcal{G}^{\mathsf{SC}}_{i \to j} = \mathcal{G}_{i_0 \to i_1} \otimes \mathcal{G}_{i_1 \to i_2} \otimes \cdots \otimes \mathcal{G}_{i_{t-1} \to i_t}. \tag{13}$$

(The definition of $\mathcal{G}^{\mathsf{SC}}_{i \to j}$ above is where our construction deviates from previous WPRG constructions [CDRST21; PV21; Hoz21].) Furthermore, define $\mathcal{G}^{\mathsf{SC}}_{i \to i} = 1 \cdot (\texttt{empty string})$, a trivial pseudodistribution over "$\{0,1\}^0$."

Note that so far, we have only considered true probability distributions. Now, however, it is time to introduce minus signs into the picture. For indices $0 \le i < j \le n$, define

$$\mathcal{E}_{i \to j} = \mathcal{U}_1 \otimes \mathcal{G}^{\mathsf{SC}}_{i+1 \to j} - \mathcal{G}^{\mathsf{SC}}_{i \to j}.$$

Finally, for each $m \in \mathbb{N}$, define

$$\mathcal{X}^{(m)} = \sum_{\ell=0}^{m} \sum_{\substack{i_0, \ldots, i_\ell \in \mathbb{N} \\ 0 \le i_0 < \cdots < i_\ell = n}} \mathcal{G}^{\mathsf{SC}}_{0 \to i_0} \otimes \mathcal{E}_{i_0 \to i_1} \otimes \mathcal{E}_{i_1 \to i_2} \otimes \cdots \otimes \mathcal{E}_{i_{\ell-1} \to i_\ell}.$$

This concludes the construction of our low-error pseudodistribution $\mathcal{X}^{(m)}$. In the next subsection, we will show that this construction "implements" the error-reduction framework that we described in Section 4.

5.2 Correspondence Between Our Pseudodistributions and Matrices

Let B be a width-w length-n ROBP that we seek to fool, with vertex set $V = V^{(0)} \cup \cdots \cup V^{(n)}$. We will interchangeably use $V^{(0)} \cup \cdots \cup V^{(n)}$ and $[(n+1)w]$. Both of them refer to the set of vertices of B. Recall that $\mathbf{B}\colon \{0,1\}^n \to \{0,1\}^{V^{(n)} \times V^{(0)}}$ is the matrix-valued function computed by B, i.e., $\mathbf{B}(x)_{v,u} = B^{v \leftarrow u}(x)$. Let $\mathbf{W}_{n \leftarrow 0} \in \mathbb{R}^{V^{(n)} \times V^{(0)}}$ be the transition matrix of B with truly random input, namely $\mathbf{W}_{n \leftarrow 0} = \mathbb{E}[\mathbf{B}]$. Let $\widetilde{\mathbf{W}}_{n \leftarrow 0}$ be the transition matrix of B with pseudorandom input generated by $\mathcal{G}$, namely $\widetilde{\mathbf{W}}_{n \leftarrow 0} = \mathbb{E}[\mathbf{B}(\mathcal{G})]$.

More generally, for $0 \le \ell < r \le n$, we define $B_{r \leftarrow \ell}$ to be the subprogram of B starting at the ℓ-th layer and ending at the r-th layer. We use $\mathbf{B}_{r \leftarrow \ell}$ to denote the corresponding function mapping $\{0,1\}^{r-\ell} \to \{0,1\}^{w \times w}$, and we let $\mathbf{W}_{r \leftarrow \ell} = \mathbb{E}[\mathbf{B}_{r \leftarrow \ell}]$ and $\widetilde{\mathbf{W}}_{r \leftarrow \ell} = \mathbb{E}[\mathbf{B}_{r \leftarrow \ell}(\mathcal{G}_{\ell \to r})]$.

[11] Different from the index for matrices, we use forward arrow $i \to j$ to index $\mathcal{G}$, because the string concatenation operation is applied from left to right.

Thus, the PRG $\mathcal{G}$ induces an approximation ensemble $\widetilde{\mathcal{W}} = \{\widetilde{\mathbf{W}}_{r\leftarrow\ell} : (\ell, r) \in E(\mathsf{SC})\}$. Using this ensemble $\widetilde{\mathcal{W}}$, we define $\widehat{\mathbf{L}}^{-1}$ and $\mathbf{A}_m$ as in Section 4. Our goal for this section is to show that the matrix $\mathbf{A}_m$ describes the effect of the pseudodistribution $\mathcal{X}^{(m)}$ on the program B. As a first step, let us analyze the effect of the distribution $\mathcal{G}^{\mathsf{SC}}_{\ell\to r}$ on the subprogram $B_{r\leftarrow\ell}$.

Lemma 5.2. *Let $0 \le \ell < r \le n$, and let $\ell = i_0 \to i_1 \to \cdots \to i_t = r$ be the shortest path from ℓ to r in* SC*. Then*

$$\mathbb{E}\left[\mathbf{B}_{r\leftarrow\ell}(\mathcal{G}^{\mathsf{SC}}_{\ell\to r})\right] = \widetilde{\mathbf{W}}_{i_t\leftarrow i_{t-1}} \cdot \widetilde{\mathbf{W}}_{i_{t-1}\leftarrow i_{t-2}} \cdots \widetilde{\mathbf{W}}_{i_1\leftarrow i_0} = \widehat{\mathbf{L}}^{-1}_{r\leftarrow\ell}.$$

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

$$\mathbf{B}_{r\leftarrow\ell}(x_{\ell\to r}) = \mathbf{B}_{i_t\leftarrow i_{t-1}}(x_{i_{t-1}\to i_t}) \cdot \mathbf{B}_{i_{t-1}\leftarrow i_{t-2}}(x_{i_{t-2}\to i_{t-1}}) \cdots \mathbf{B}_{i_1\leftarrow i_0}(x_{i_0\to i_1}),$$

where $x_{i\to j} = (x_{i+1}, x_{i+2}, \ldots, x_j)$ and $\cdot$ is matrix multiplication. Furthermore, $\mathcal{G}^{\mathsf{SC}}_{\ell\to r}$ is a product distribution $\mathcal{G}_{i_0\to i_1} \otimes \cdots \otimes \mathcal{G}_{i_{t-1}\to i_t}$. The expected value of a product of independent random variables is the product of expectations, so

$$\begin{aligned}\mathbb{E}\left[\mathbf{B}_{r\leftarrow\ell}(\mathcal{G}^{\mathsf{SC}}_{\ell\to r})\right] &= \mathbb{E}\left[\mathbf{B}_{i_t\leftarrow i_{t-1}}(\mathcal{G}_{i_{t-1}\to i_t})\right] \cdot \mathbb{E}\left[\mathbf{B}_{i_{t-1}\leftarrow i_{t-2}}(\mathcal{G}_{i_{t-2}\to i_{t-1}})\right] \cdots \mathbb{E}\left[\mathbf{B}_{i_1\leftarrow i_0}(\mathcal{G}_{i_0\to i_1})\right] \\ &= \widetilde{\mathbf{W}}_{i_t\leftarrow i_{t-1}} \cdot \widetilde{\mathbf{W}}_{i_{t-1}\leftarrow i_{t-2}} \cdots \widetilde{\mathbf{W}}_{i_1\leftarrow i_0}.\end{aligned}$$

□

Next, let us analyze the effect of the pseudodistribution $\mathcal{E}_{\ell\to r}$ on the subprogram $B_{r\leftarrow\ell}$. To be more precise, we will give a formula for the *pseudoexpectation* of the subprogram, defined next.

Definition 5.3 (Pseudoexpectation)**.** *Let $w, n \in \mathbb{N}$, let $\mathbf{F}\colon \{0,1\}^n \to \mathbb{R}^{w\times w}$ be a function, and let $\mathcal{X}$ be a pseudodistribution over $\{0,1\}^n$, say $\mathcal{X} = \sum_{i=1}^R \lambda_i \cdot x^{(i)}$. We define the* pseudoexpectation *of $\mathbf{F}$ under $\mathcal{X}$ by the formula*

$$\widetilde{\mathbb{E}}[\mathbf{F}(\mathcal{X})] = \sum_{i=1}^R \lambda_i \cdot \mathbf{F}(x^{(i)}) \in \mathbb{R}^{w\times w}.$$

One can show that pseudoexpectation satisfies the following two properties, analogous to familiar facts about the standard expectation operator.

Lemma 5.4 (Pseudoexpectation under mixture equals mixture of pseudoexpectations)**.** *Let $w, n \in \mathbb{N}$, let $\mathbf{F}\colon \{0,1\}^n \to \mathbb{R}^{w\times w}$ be a function, let $\mathcal{X}, \mathcal{Y}$ be pseudodistributions over $\{0,1\}^n$, and let $c \in \mathbb{R}$. Then*

$$\widetilde{\mathbb{E}}[\mathbf{F}(\mathcal{X} + c\mathcal{Y})] = \widetilde{\mathbb{E}}[\mathbf{F}(\mathcal{X})] + c \cdot \widetilde{\mathbb{E}}[\mathbf{F}(\mathcal{Y})].$$

Lemma 5.5 (Pseudoexpectation of product equals product of pseudoexpectations, assuming "independence")**.** *Let $w, n_0, n_1 \in \mathbb{N}$, let $\mathbf{F}_b\colon \{0,1\}^{n_b} \to \mathbb{R}^{w\times w}$ be a function for each $b \in \{0,1\}$, let $\mathbf{F}(x,y) = \mathbf{F}_0(x) \cdot \mathbf{F}_1(y)$, and let $\mathcal{X}$ and $\mathcal{Y}$ be pseudodistributions over $\{0,1\}^{n_0}$ and $\{0,1\}^{n_1}$ respectively. Then*

$$\widetilde{\mathbb{E}}[\mathbf{F}(\mathcal{X} \otimes \mathcal{Y})] = \widetilde{\mathbb{E}}[\mathbf{F}_0(\mathcal{X})] \cdot \widetilde{\mathbb{E}}[\mathbf{F}_1(\mathcal{Y})].$$

The effect of $\mathcal{E}_{\ell\to r}$ on $B_{r\leftarrow\ell}$ is given by the following lemma.

Lemma 5.6. *The matrix $\widetilde{\mathbb{E}}\left[\mathbf{B}_{r\leftarrow\ell}(\mathcal{E}_{\ell\to r})\right] \in \mathbb{R}^{w\times w}$ is equal to the (r,ℓ)-th block of $\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L} \in \mathbb{R}^{(n+1)w\times(n+1)w}$.*

Proof. By Lemma 5.4, Lemma 5.5, and the definition of $\mathcal{E}_{\ell\to r}$, we have

$$\begin{aligned}\widetilde{\mathbb{E}}\left[\mathbf{B}_{r\leftarrow\ell}(\mathcal{E}_{\ell\to r})\right] &= \mathbb{E}\left[\mathbf{B}_{r\leftarrow\ell+1}(\mathcal{G}^{\mathsf{SC}}_{\ell+1\to r})\right]\cdot\mathbb{E}\left[\mathbf{B}_{\ell+1\leftarrow\ell}\right] - \mathbb{E}\left[\mathbf{B}_{r\leftarrow\ell}(\mathcal{G}^{\mathsf{SC}}_{\ell\to r})\right]\\ &= \widehat{\mathbf{L}}^{-1}_{r,\ell+1}\cdot\mathbf{W}_{\ell+1,\ell} - \widehat{\mathbf{L}}^{-1}_{r,\ell}\end{aligned}$$

by Lemma 5.2. The (k,ℓ)-th block of $\mathbf{W}$ is zero whenever $k\neq\ell+1$, so the expression above is the (r,ℓ)-th block of $\widehat{\mathbf{L}}^{-1}\cdot\mathbf{W}-\widehat{\mathbf{L}}^{-1}$. Furthermore, since $\ell<r$, it can equally well be considered to be the (r,ℓ)-th block of $\mathbf{I}+\widehat{\mathbf{L}}^{-1}\cdot\mathbf{W}-\widehat{\mathbf{L}}^{-1}$. Finally,

$$\mathbf{I}+\widehat{\mathbf{L}}^{-1}\cdot\mathbf{W}-\widehat{\mathbf{L}}^{-1} = \mathbf{I}+\widehat{\mathbf{L}}^{-1}(\mathbf{W}-\mathbf{I}) = \mathbf{I}-\widehat{\mathbf{L}}^{-1}\mathbf{L}. \qquad \square$$

The matrix $\mathbf{I}-\widehat{\mathbf{L}}^{-1}\mathbf{L}$ that appears in Lemma 5.6 is the same error matrix that appears in Richardson iteration. Recall that we defined the matrix $\mathbf{A}_m$ in Section 4 by performing m rounds of Richardson iteration:

$$\mathbf{A}_m = \sum_{i=0}^{m}(\mathbf{I}-\widehat{\mathbf{L}}^{-1}\mathbf{L})^i\cdot\widehat{\mathbf{L}}^{-1}.$$

The matrix $\mathbf{A}_m$ "corresponds to" the pseudodistribution $\mathcal{X}^{(m)}$ in the following sense.

Claim 5.7 (Pseudodistribution-matrix correspondence)**.** *The matrix* $\widetilde{\mathbb{E}}\left[\mathbf{B}(\mathcal{X}^{(m)})\right]$ *is equal to the* $(n,0)$ *block of* $\mathbf{A}_m\in\mathbb{R}^{(n+1)w\times(n+1)w}$.

Proof. By Lemma 5.4, Lemma 5.5, Lemma 5.2, Lemma 5.6, and the definition of $\mathcal{X}^{(m)}$, we have

$$\widetilde{\mathbb{E}}\left[\mathbf{B}(\mathcal{X}^{(m)})\right] = \sum_{\ell=0}^{m}\sum_{\substack{i_0,\dots,i_\ell\in\mathbb{N}\\ 0\le i_0<\cdots<i_\ell=n}}(\mathbf{I}-\widehat{\mathbf{L}}^{-1}\mathbf{L})_{i_\ell,i_{\ell-1}}\cdots(\mathbf{I}-\widehat{\mathbf{L}}^{-1}\mathbf{L})_{i_1,i_0}\cdot\widehat{\mathbf{L}}^{-1}_{i_0,0}.$$

This is precisely the $(n,0)$ block of $\mathbf{A}_m$, because $\widehat{\mathbf{L}}^{-1}$ and $\mathbf{L}$ are lower triangular with $\mathbf{I}$ in the diagonal blocks. $\square$

5.3 Template for Constructing WPRGs

We conclude Section 5 by giving a brief outline of how we will use our framework in future sections to construct low-error WPRGs. Fix a class $\mathcal{B}$ of ROBPs that we want to fool.

Identifying an appropriate moderate-error PRG. To construct a low-error WPRG for $\mathcal{B}$, we will begin by identifying a PRG $\mathcal{G}\colon\{0,1\}^s\to\{0,1\}^n$. Then, we will show that for every $B\in\mathcal{B}$, we have

$$\|\mathbf{L}^{-1}\Delta\mathbf{W}\|\le\delta, \tag{14}$$

where $\mathbf{L}$ is the Laplacian matrix of B, $\Delta\mathbf{W}$ is the correction graph transition matrix induced by B and $\mathcal{G}$, $\|\cdot\|$ is an appropriately-chosen extended submultiplicative matrix seminorm, and δ is "moderately small."

Decreasing the error via our error reduction framework. The bound (14) implies that our error reduction framework is applicable. As explained earlier in this section, we convert $\mathcal{G}$ into a pseudodistribution $\mathcal{X}^{(m)}$. This pseudodistribution which corresponds to the matrix $\mathbf{A}_m$ from Section 4. As explained in Section 4, the matrix $\mathbf{A}_m$ has low error, in the sense that

$$\|\mathbf{I} - \mathbf{A}_m\mathbf{L}\| \leq (2\delta)^{(m+1)}.$$

Using properties of the specific extended seminorm $\|\cdot\|$, we will show as a consequence that $\mathcal{X}^{(m)}$ actually fools the Boolean function B with low error. (This corresponds to analyzing $\mathbf{A}_m - \mathbf{L}^{-1}$ rather than $\mathbf{I} - \mathbf{A}_m\mathbf{L}$.)

Derandomizing $\mathcal{X}^{(m)}$ with correlated seeds. At this point, we will have a WPRG that fools $\mathcal{B}$ with low error. However, looking again at the pseudodistribution $\mathcal{X}^{(m)}$, it is a sum of $M = n^{O(m)}$ signed terms, each of which is a tensor product of up to $O(m \log n)$ copies of $\mathcal{G}$ (and up to m copies of $\mathcal{U}_1$). Sampling even one such term would cost $O(m \cdot s \cdot \log n)$ truly random bits, which we cannot afford.

The final step of the WPRG constructions will be to modify $\mathcal{X}^{(m)}$ by using *correlated seeds* across the different copies of $\mathcal{G}$. This step is known already from previous work [CDRST21; PV21]. Since the arguments differ slightly between the case of bounded-width regular ROBPs and the case of unbounded-width permutation ROBPs, we present the details in future sections.

6 WPRG for Regular ROBPs

In this section, we present our WPRG for regular ROBPs (Theorem 1.7). Actually, we will prove a theorem that is stronger than Theorem 1.7 in two respects. First, we will consider a more general class of branching programs. We construct a WPRG that fools any standard-order ROBP in which all subprograms have low *weight* as defined by Braverman, Rao, Raz, and Yehudayoff [BRRY14].

Definition 6.1 (Weight of an ROBP [BRRY14])**.** *Let B be a standard-order ROBP. The* weight *of an edge $e = (u, v)$ is defined by* $\mathsf{Weight}(e) = |\mathbb{E}[B^{\leftarrow u}] - \mathbb{E}[B^{\leftarrow v}]|$*, i.e.,* $\mathsf{Weight}(e)$ *is the absolute difference between the acceptance probabilities when we start at u and when we start at v. The weight of the program, denoted* $\mathsf{Weight}(B)$*, is the sum of the weights of all edges.*

Second, we will prove that our WPRG has the following *boundedness* property, introduced by Chattopadhyay and Liao [CL20] (modifying a prior definition by Braverman, Cohen, and Garg [BCG20]).

Definition 6.2 (K-bounded WPRG [CL20])**.** *Let $(\mathcal{G}, \mu)$ be a WPRG and let $K > 0$. We say that $(\mathcal{G}, \mu)$ is* K-bounded *if $|\mu(u)| \leq K$ for every seed u.*

Theorem 6.3 (WPRG for ROBPs in which all subprograms have low weight)**.** *For all $n, w, W \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit ε-WPRG with seed length*

$$\widetilde{O}\left(\log n \cdot \left(\log(W \cdot w) + \sqrt{\log(1/\varepsilon)}\right) + \log(1/\varepsilon)\right)$$

that ε-fools every width-w length-n standard-order ROBP B that satisfies the following property: For every vertex v, we have $\mathsf{Weight}(B^{v\leftarrow}) \leq W$*. Furthermore, the WPRG is K-bounded for a value $K = n^{O\left(\sqrt{\log(1/\varepsilon)}\right)}$.*

Theorem 6.3 implies Theorem 1.7, because Braverman, Rao, Raz, and Yehudayoff showed that for every width-w length-n standard-order regular ROBP B and every vertex v, we have $\mathsf{Weight}(B^{v\leftarrow}) \leq O(w)$ [BRRY14]. The greater generality of Theorem 6.3 will be essential for our treatment of width-3 ROBPs in Section 7.

Proof outline. The proof of Theorem 6.3 is based on the error-reduction procedure from Section 5. Denote by $\mathcal{B}$ the class of ROBPs satisfying the property stated in Theorem 6.3. Let $\mathcal{G}\colon \{0,1\}^s \to \{0,1\}^n$ be a PRG that τ-fools every program $B \in \mathcal{B}$, where

$$\varepsilon < \tau < \frac{1}{\mathrm{poly}(W, w, \log n)}.$$

We will show how to convert $\mathcal{G}$ into an ε-WPRG with seed length

$$\widetilde{O}\left(s + \frac{\log(w/\varepsilon) \cdot \log n}{\log(1/\tau)} + \log(w/\varepsilon)\right).$$

To conclude, we will choose $\tau \approx 2^{-\sqrt{\log(1/\varepsilon)}}$ and take $\mathcal{G}$ to be the BRRY PRG [BRRY14]. It might be helpful to keep in mind the case $w = O(1)$, $W = O(1)$, $\varepsilon = 1/n$, and $\tau = 2^{-\sqrt{\log n}}$.

To construct the low-error WPRG, we will follow the framework presented in Section 5. In Section 6.1, we show that the matrix $\mathbf{L}^{-1}\Delta\mathbf{W}$ has a moderately small norm, as required by our error reduction framework. In Section 6.2, we construct a WPRG with a low error but with a relatively large seed. Finally, in Section 6.3, we replace independent seeds with correlated seeds to get a WPRG with a low error and a low seed length. (This last step is the same as in prior works [CDRST21; PV21].)

6.1 Bounding $\|\mathbf{L}^{-1}\Delta\mathbf{W}\|$

For convenience, assume that the first output bit of the PRG $\mathcal{G}$ is perfectly uniform. (If this property does not hold, $\mathcal{G}$ can be modified to have this property while increasing the seed length by only one bit.) Our goal in this subsection is to prove the following lemma.

Lemma 6.4 (Moderate-error bound when all subprograms have low weight)**.** *Let $B \in \mathcal{B}$. Let $\mathbf{L}$ be the Laplacian matrix of B, and let $\Delta\mathbf{W}$ be the correction graph transition matrix induced by B and $\mathcal{G}$ as explained in Section 4 and Section 5. Then*

$$\|\mathbf{L}^{-1}\Delta\mathbf{W}\|_\infty \leq 1.5 \cdot \tau \cdot W \cdot w^2 \cdot \log n.$$

In Lemma 6.4, the notation $\|\cdot\|_\infty$ denotes the matrix norm induced by the vector ℓ_∞ norm, defined as

$$\|\mathbf{M}\|_\infty = \max_{z:\|z\|_\infty=1}\{\|\mathbf{M}z\|_\infty\} = \max_{i\text{: row of }\mathbf{M}}\left\{\sum_{j\text{: column of }\mathbf{M}} |\mathbf{M}_{i,j}|\right\}.$$

This matrix norm is submultiplicative, so as explained in Section 4 and Section 5, bounding $\|\mathbf{L}^{-1}\Delta\mathbf{W}\|_\infty$ will enable us to use our error reduction framework.

The first step in the proof of Lemma 6.4 is to show (roughly speaking) that the PRG $\mathcal{G}$ fools subprograms with error significantly *less* than τ when we have a very low weight bound (much less than 1). To be more precise, we use the following notation.

Definition 6.5 (Weight of a region in the ROBP)**.** *Let B be a standard-order ROBP with layers $V^{(0)}, \dots, V^{(n)}$. For $i \in [n]$, let $E^{(i)}$ be the set of incoming edges to vertices in $V^{(i)}$, i.e., $E^{(i)} = E(B) \cap (V^{(i-1)} \times V^{(i)})$. Let $0 \le i \le j \le n$. We define $\mathsf{Weight}(B, i, j)$ to be the weight of all edges between $V^{(i)}$ and $V^{(j)}$, i.e.,*

$$\mathsf{Weight}(B, i, j) = \sum_{e \in E^{(i+1)} \cup \cdots \cup E^{(j)}} \mathsf{Weight}(e).$$

Lemma 6.6. *Let B be a standard-order ROBP on vertices $V^{(0)} \cup \cdots \cup V^{(n)}$. Let $0 \le \ell < r \le j \le n$, let $u \in V^{(\ell)}$, let $q \in V^{(j)}$, and let $\mathcal{Y}$ and $\mathcal{U}$ be independent random variables, where $\mathcal{Y} \in \{0,1\}^{r-\ell}$ and $\mathcal{U}$ is distributed uniformly over $\{0,1\}^{j-r}$. Assume that for every $v \in V^{(r)}$, the distribution $\mathcal{Y}$ fools the subprogram $B^{v \leftarrow u}$ with error α. Then the concatenation $(\mathcal{Y}, \mathcal{U})$ fools $B^{q \leftarrow u}$ with error $\alpha \cdot w \cdot \mathsf{Weight}(B^{q \leftarrow u}, \ell, r)/2$.*

Proof. Let S be the set of vertices $v \in V^{(r)}$ that are reachable from u. Braverman, Rao, Raz, and Yehudayoff showed [BRRY14, Proof of Proposition 5] that for every $v, v' \in S$, we have

$$|\mathbb{E}[B^{q \leftarrow v}] - \mathbb{E}[B^{q \leftarrow v'}]| \le \mathsf{Weight}(B^{q \leftarrow u}, \ell, r).$$

Therefore, there is a value η such that for every vertex $v \in S$, we have

$$|\mathbb{E}[B^{q \leftarrow v}] - \eta| \le \frac{1}{2}\mathsf{Weight}(B^{q \leftarrow u}, \ell, r).$$

Define $\delta_v = \mathbb{E}[B^{q \leftarrow v}] - \eta$. Then

$$\begin{aligned}
&|\mathbb{E}[B^{q \leftarrow u}(0^\ell, \mathcal{Y}, \mathcal{U}, 0^{n-j})] - \mathbb{E}[B^{q \leftarrow u}]| \\
&\quad = \left| \sum_{v \in S} (\mathbb{E}[B^{v \leftarrow u}(\mathcal{Y})] - \mathbb{E}[B^{v \leftarrow u}]) \cdot (\eta + \delta_v) \right| \\
&\quad \le \eta \cdot \left| \sum_{v \in S} \mathbb{E}[B^{v \leftarrow u}(\mathcal{Y})] - \sum_{v \in S} \mathbb{E}[B^{v \leftarrow u}] \right| + \sum_{v \in S} |\mathbb{E}[B^{v \leftarrow u}(\mathcal{Y})] - \mathbb{E}[B^{v \leftarrow u}]| \cdot |\delta_v| \\
&\quad = \eta \cdot |1 - 1| + \sum_{v \in S} |\mathbb{E}[B^{v \leftarrow u}(\mathcal{Y})] - \mathbb{E}[B^{v \leftarrow u}]| \cdot \mathsf{Weight}(B^{q \leftarrow u}, \ell, r)/2 \\
&\quad \le \alpha \cdot w \cdot \mathsf{Weight}(B^{q \leftarrow u}, \ell, r)/2.
\end{aligned}$$

□

With Lemma 6.6 in hand, we now turn to analyzing the matrix $\mathbf{L}^{-1}\Delta\mathbf{W}$. As suggested in Section 4, we decompose $\Delta\mathbf{W}$ as $\Delta\mathbf{W} = \Delta\mathbf{W}^{(0)} + \cdots + \Delta\mathbf{W}^{(\log n)}$, where $\Delta\mathbf{W}^{(t)}_{j \to i}$ is nonzero only for edges $(i, j) \in E(\mathsf{SC})$ where $j = i + 2^t$. Because the first bit of the output of $\mathcal{G}$ is perfectly uniform, we have $\Delta\mathbf{W}^{(0)} = 0$, so our decomposition becomes $\Delta\mathbf{W} = \Delta\mathbf{W}^{(1)} + \cdots + \Delta\mathbf{W}^{(\log n)}$. For a fixed $t \in [\log n]$, let us bound each individual entry of the matrix $\mathbf{L}^{-1}\Delta\mathbf{W}^{(t)}$.

Claim 6.7 (Bound on one entry of $\mathbf{L}^{-1}\Delta\mathbf{W}^{(t)}$)**.** *Let $B \in \mathcal{B}$, let $\mathbf{L}$ be the Laplacian matrix of B, and let $\Delta\mathbf{W}$ be the correction graph transition matrix induced by B and $\mathcal{G}$. Let $t \in [\log n]$, and define $\Delta\mathbf{W}^{(t)}$ as above. Let $0 \le \ell < r \le j \le n$, where $\ell \in U_t$ and $r = \ell + 2^t$. Let $u \in V^{(\ell)}$ and $q \in V^{(j)}$. Then*

$$\left| \left(\mathbf{L}^{-1}\Delta\mathbf{W}^{(t)} \right)_{q,u} \right| \le 1.5 \cdot \tau \cdot w \cdot \mathsf{Weight}(B^{q \leftarrow u}, \ell, r).$$

Note that the claim above assumes that $\ell \in U_t$. (Recall from Definition 4.1 that U_t is the set of multiples of 2^t between 0 and n.) If $\ell \notin U_t$, then it is easy to see that $\left(\mathbf{L}^{-1}\Delta\mathbf{W}^{(t)} \right)_{q,u} = 0$.

Proof. Working through the definitions, we have

$$\begin{aligned}\left(\mathbf{L}^{-1}\Delta\mathbf{W}^{(t)}\right)_{q,u} &= \left(\mathbf{W}_{j\leftarrow r}\cdot\left(\widetilde{\mathbf{W}}_{r\leftarrow\ell}-\widetilde{\mathbf{W}}_{r\leftarrow\frac{\ell+r}{2}}\cdot\widetilde{\mathbf{W}}_{\frac{\ell+r}{2}\leftarrow\ell}\right)\right)_{q,u}\\ &= \mathbb{E}[B^{q\leftarrow u}(\mathcal{Y},\mathcal{U})]-\mathbb{E}[B^{q\leftarrow u}(\mathcal{Y}',\mathcal{Y}'',\mathcal{U})],\end{aligned}$$

where $\mathcal{Y}$ is distributed as the first 2^t bits of a sample from $\mathcal{G}$; each of $\mathcal{Y}'$ and $\mathcal{Y}''$ is distributed as the first 2^{t-1} bits of a sample from $\mathcal{G}$; the variable $\mathcal{U}$ is uniformly distributed over $\{0,1\}^{j-r}$; and $\mathcal{Y}$, $\mathcal{Y}'$, $\mathcal{Y}''$, and $\mathcal{U}$ are independent. By Lemma 6.6 with $\alpha=\tau$, we have

$$|\mathbb{E}[B^{q\leftarrow u}(\mathcal{Y},\mathcal{U})]-\mathbb{E}[B^{q\leftarrow u}]|\le 0.5\cdot\tau\cdot w\cdot\mathsf{Weight}(B^{q\leftarrow u},\ell,r).$$

Similarly, one can show that the concatenation $(\mathcal{Y}',\mathcal{Y}'')$ fools subprograms $B_{v\leftarrow u}$ with error 2τ, so we may apply Lemma 6.6 again, this time with $\alpha=2\tau$, to get

$$|\mathbb{E}[B^{q\leftarrow u}(\mathcal{Y}',\mathcal{Y}'',\mathcal{U})]-\mathbb{E}[B^{q\leftarrow u}]|\le \tau\cdot w\cdot\mathsf{Weight}(B^{q\leftarrow u},\ell,r).$$

The triangle inequality completes the proof. □

Summing everything up completes the analysis of $\mathbf{L}^{-1}\Delta\mathbf{W}$:

Proof of Lemma 6.4. For each fixed $t\in[\log n]$, by Claim 6.7 we have

$$\left\|\mathbf{L}^{-1}\Delta\mathbf{W}^{(t)}\right\|_\infty\le\max_q\sum_u\left|\left(\mathbf{L}^{-1}\Delta\mathbf{W}^{(t)}\right)_{q,u}\right|\le\max_q\sum_{\ell\in U_t}\sum_{u\in V^{(\ell)}}1.5\cdot\tau\cdot w\cdot\mathsf{Weight}(B^{q\leftarrow u},\ell,\ell+2^t).$$

The value $\mathsf{Weight}(B^{q\leftarrow u},\ell,\ell+2^t)$ does not depend on the start vertex $u\in V^{(\ell)}$. Therefore,

$$\begin{aligned}\left\|\mathbf{L}^{-1}\Delta\mathbf{W}^{(t)}\right\|_\infty &\le\max_q 1.5\cdot\tau\cdot w^2\cdot\sum_{\ell\in U_t}\mathsf{Weight}(B^{q\leftarrow},\ell,\ell+2^t)=\max_q 1.5\cdot\tau\cdot w^2\cdot\mathsf{Weight}(B^{q\leftarrow})\\ &\le 1.5\cdot\tau\cdot w^2\cdot W.\end{aligned}$$

Finally, summing over all t, we have

$$\|\mathbf{L}^{-1}\Delta\mathbf{W}\|_\infty\le\sum_{t=1}^{\log n}\left\|\mathbf{L}^{-1}\Delta\mathbf{W}^{(t)}\right\|_\infty\le 1.5\cdot\tau\cdot w^2\cdot W\cdot\log n.$$ □

6.2 Low-Error High-Seed-Length WPRG: Independent Seeds

In the previous subsection, we showed a bound on $\|\mathbf{L}^{-1}\Delta\mathbf{W}\|$. By the results in Section 4 and Section 5, this implies that our error reduction framework works, and hence the pseudodistribution $\mathcal{X}^{(m)}$ fools $\mathcal{B}$ with low error. In particular, we get the following.

Claim 6.8 ($\mathcal{X}^{(m)}$ fools $\mathcal{B}$ with low error)**.** *Suppose $\tau\le\frac{1}{9w^4\cdot W^2\cdot\log^2 n}$. Then for every $B\in\mathcal{B}$ and $m\in\mathbb{N}$,*

$$\left|\widetilde{\mathbb{E}}\left[B(\mathcal{X}^{(m)})\right]-\mathbb{E}[B]\right|\le\tau^{(m+1)/2}\cdot w.$$

Proof. By Lemma 6.4 and Theorem 4.4, we have

$$\|\mathbf{I}-\mathbf{A}_m\mathbf{L}\|_\infty\le(3\tau\cdot w^2\cdot W\cdot\log n)^{m+1}\le\tau^{(m+1)/2}.$$

Therefore, for each basis vector e_u, we have

$$\|\mathbf{A}_m e_u - \mathbf{L}^{-1} e_u\|_\infty \le \|\mathbf{A}_m \mathbf{L}\mathbf{L}^{-1} e_u - \mathbf{L}^{-1} e_u\|_\infty \le \tau^{(m+1)/2} \cdot \|\mathbf{L}^{-1} e_u\|_\infty \le \tau^{(m+1)/2},$$

since each entry of $\mathbf{L}^{-1} e_u$ is a probability. Consequently, each entry of $\mathbf{A}_m$ differs from the corresponding entry of $\mathbf{L}^{-1}$ by at most $\tau^{(m+1)/2}$. By Claim 5.7, it follows that for each vertex $v \in V^{(n)}$, we have

$$\left| \widetilde{\mathbb{E}} \left[B^{v\leftarrow}(\mathcal{X}^{(m)}) \right] - \mathbb{E}[B^{v\leftarrow}] \right| \le \tau^{(m+1)/2}.$$

Summing over at most w accepting vertices completes the proof. □

6.3 Final WPRG Construction: Correlated Seeds

At this point, we have shown that the pseudodistribution $\mathcal{X}^{(m)}$ fools $\mathcal{B}$ with low error. However, as we mentioned in Section 5.3, sampling $\mathcal{X}^{(m)}$ directly requires a larger seed length than we can afford. In this subsection, following prior work [CDRST21; PV21], we show how to reduce the seed length by using the Impagliazzo-Nisan-Wigderson (INW) PRG [INW94] to generate a sequence of correlated seeds to $\mathcal{G}$.

The final WPRG construction. Recall from Section 5 that $\mathcal{X}^{(m)}$ is defined by the formula

$$\mathcal{X}^{(m)} = \sum_{\ell=0}^{m} \sum_{\substack{i_0,\dots,i_\ell \in \mathbb{N} \\ 0 \le i_0 < \cdots < i_\ell = n}} \mathcal{G}^{\mathsf{SC}}_{0\to i_0} \otimes \mathcal{E}_{i_0 \to i_1} \otimes \mathcal{E}_{i_1 \to i_2} \otimes \cdots \otimes \mathcal{E}_{i_{\ell-1} \to i_\ell},$$

where $\mathcal{E}_{i\to j}$ is given by

$$\mathcal{E}_{i\to j} = \mathcal{U}_1 \otimes \mathcal{G}^{\mathsf{SC}}_{i+1\to j} - \mathcal{G}^{\mathsf{SC}}_{i\to j}$$

and $\mathcal{G}^{\mathsf{SC}}_{\ell\to r}$ is the tensor product of $\mathcal{G}_{i\to j}$ over all edges (i,j) in the shortest path from ℓ to r through SC_n. This path has length $O(\log n)$. The PRG $\mathcal{G}$ has seed length s, and a seed of length s is also trivially sufficient for sampling a single uniform random bit ($\mathcal{U}_1$). Therefore, by expanding each $\mathcal{E}_{i\to j}$ and each $\mathcal{G}^{\mathsf{SC}}_{\ell\to r}$, we can write $\mathcal{X}^{(m)}$ in the form

$$\mathcal{X}^{(m)} = \sum_{i=1}^{K} \sigma_i \cdot \sum_{y^{(1)},\dots,y^{(r)} \in \{0,1\}^s} 2^{-sr} \cdot \mathcal{G}_{i,1}(y^{(1)}) \circ \cdots \circ \mathcal{G}_{i,r}(y^{(r)}),$$

where $K = n^{O(m)}$, $r = O(m \log n)$, $\sigma_i \in \{-1,+1\}$, $\mathcal{G}_{i,j}$ is a function $\mathcal{G}_{i,j} : \{0,1\}^s \to \{0,1\}^{n_{i,j}}$ for some $0 \le n_{i,j} \le n$, and $\circ$ denotes string concatenation. For convenience, we will assume that K is a power of two; this can be accomplished by adding dummy terms with $\sigma_i = 0$.

Let $\mathcal{Y}$ be a distribution over $(\{0,1\}^s)^r$ that γ-fools width-w standard-order ROBPs over the alphabet $\{0,1\}^s$, where $\gamma = \frac{\varepsilon}{2K}$. Using the INW PRG [INW94], we can explicitly sample $\mathcal{Y}$ using a seed of length $q = O(s + \log(wr/\gamma) \cdot \log r)$. Write $\mathcal{Y}$ using pseudodistribution notation as

$$\mathcal{Y} = \sum_{z \in \{0,1\}^q} 2^{-q} \cdot y_z^{(1)} \circ \cdots \circ y_z^{(r)}$$

where $y_z^{(j)} \in \{0,1\}^s$ for each $j \in [r]$. Our final pseudodistribution $\mathcal{Z}$ over $\{0,1\}^n$ is given by the formula

$$\mathcal{Z} = \sum_{i=1}^{K} \sigma_i \cdot \sum_{z \in \{0,1\}^q} 2^{-q} \cdot \mathcal{G}_{i,1}(y_z^{(1)}) \circ \cdots \circ \mathcal{G}_{i,r}(y_z^{(r)}).$$

This pseudodistribution $\mathcal{Z}$ corresponds to the WPRG $(\mathcal{G}', \mu)$, where $\mathcal{G}' \colon [K] \times \{0,1\}^q \to \{0,1\}^n$ is given by

$$\mathcal{G}'(i, z) = \mathcal{G}_{i,1}(y_z^{(1)}) \circ \cdots \circ \mathcal{G}_{i,r}(y_z^{(r)})$$

and $\mu \colon [K] \times \{0,1\}^q \to \mathbb{R}$ is given by $\mu(i, z) = K \cdot \sigma_i$.

Error and seed length. The following claim shows that using correlated seeds only increases the error by $\varepsilon/2$ compared to independent seeds.

Claim 6.9. *For every $B \in \mathcal{B}$,*

$$\left| \widetilde{\mathbb{E}} \left[B(\mathcal{Z}) \right] - \widetilde{\mathbb{E}} \left[B(\mathcal{X}^{(m)}) \right] \right| \leq \frac{\varepsilon}{2}.$$

Proof. For each $i \in [K]$, define $f_i \colon (\{0,1\}^s)^r \to \{0,1\}$ by

$$f_i(y^{(1)}, \ldots, y^{(r)}) = B(\mathcal{G}_{i,1}(y^{(1)}) \circ \cdots \circ \mathcal{G}_{i,r}(y^{(r)})).$$

Then

$$\begin{aligned} \left| \widetilde{\mathbb{E}} \left[B(\mathcal{Z}) \right] - \widetilde{\mathbb{E}} \left[B(\mathcal{X}^{(m)}) \right] \right| &= \left| \sum_{i=1}^{K} \sigma_i \cdot (\mathbb{E}[f_i(\mathcal{U}_{sr})] - \mathbb{E}[f_i(\mathcal{Y})]) \right| \\ &\leq \sum_{i=1}^{K} |\mathbb{E}[f_i(\mathcal{U}_{sr})] - \mathbb{E}[f_i(\mathcal{Y})]| \\ &\leq K \cdot \gamma = \varepsilon/2, \end{aligned}$$

where the last inequality holds because f_i can be computed by a width-w standard-order ROBP over the alphabet $\{0,1\}^s$. □

We choose $m = \Theta(\frac{\log(w/\varepsilon)}{\log(1/\tau)})$. That way, by Claim 6.8, $\mathcal{X}^{(m)}$ fools $\mathcal{B}$ with error $\varepsilon/2$, and hence by Claim 6.9, our final WPRG $(\mathcal{G}', \mu)$ fools $\mathcal{B}$ with error ε. The final WPRG's seed length is

$$\begin{aligned} \log K + q &= O(s + \log K + \log(wr/\gamma) \cdot \log r) \\ &= \widetilde{O}(s + m \log n + \log(w/\varepsilon)) \\ &= \widetilde{O}\left(s + \frac{\log(w/\varepsilon) \cdot \log n}{\log(1/\tau)} + \log(w/\varepsilon) \right). \end{aligned}$$

We take $\mathcal{G}$ to be the BRRY PRG [BRRY14], so $s = \widetilde{O}(\log n \cdot \log(W \cdot w/\tau))$. Furthermore, we choose

$$\tau = \min \left\{ 2^{-\sqrt{\log(1/\varepsilon)}}, \frac{1}{9w^4 \cdot W^2 \cdot \log^2 n} \right\}.$$

Therefore, the overall seed length becomes

$$\widetilde{O}\left(\log n \cdot \left(\log(W \cdot w) + \sqrt{\log(1/\varepsilon)} \right) + \log(1/\varepsilon) \right).$$

Finally, observe that $(\mathcal{G}', \mu)$ is clearly K-bounded, and recall that $K = n^{O(m)}$ and

$$m = \Theta\left(\frac{\log w}{\log(1/\tau)} + \frac{\log(1/\varepsilon)}{\log(1/\tau)} \right) \leq O\left(\frac{\log w}{\log w} + \frac{\log(1/\varepsilon)}{\sqrt{\log(1/\varepsilon)}} \right) = O\left(\sqrt{\log(1/\varepsilon)} \right),$$

completing the proof of Theorem 6.3.

7 WPRG for General Width-3 ROBP

In this section, we revisit the work of Meka, Reingold, and Tal [MRT19] for general width-3 standard-order ROBPs and improve it in several ways. Meka, Reingold, and Tal's work is based on studying the effect of pseudorandom *restrictions* on width-3 ROBPs [MRT19].

Definition 7.1 (Restrictions). *A* restriction *is a string $\rho \in \{0,1,\star\}^n$. If $\rho, \rho' \in \{0,1,\star\}^n$, then the* composition $\rho \circ \rho' \in \{0,1,\star\}^n$ *is defined as follows:*

$$(\rho \circ \rho')_i = \begin{cases} \rho_i & \text{if } \rho'_i = \star \\ \rho'_i & \text{if } \rho'_i \in \{0,1\}. \end{cases}$$

In particular, if $\rho \in \{0,1,\star\}^n$ and $x \in \{0,1\}^n$, then $x \circ \rho$ is defined by using x to fill in the $\star$ positions of ρ. If B is a function on $\{0,1\}^n$ and $\rho \in \{0,1,\star\}^n$, then the restricted function $B|_\rho$ *is another function on $\{0,1\}^n$ defined by $B|_\rho(x) = B(x \circ \rho)$.*[12]

Our main technical result in this section, which will imply our WPRG for width-3 standard-order ROBPs (Theorem 1.5), is the following.

Theorem 7.2 (Pseudorandom restrictions for width-3 ROBPs). *For every $n \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit restriction generator $\mathcal{R}\colon \{0,1\}^s \to \{0,1,\star\}^n$ with seed length $s = \widetilde{O}(\log(n/\varepsilon))$ such that for every width-3 standard-order ROBP B, if we sample $\rho = \mathcal{R}(\mathcal{U}_s)$, then*

1. *The restriction ρ* preserves the expectation *of B up to error ε. That is,*

$$|\mathbb{E}_{\rho,\mathcal{U}}[B|_\rho(\mathcal{U})] - \mathbb{E}[B]| \le \varepsilon,$$

   where $\mathcal{U} \in \{0,1\}^n$ is sampled uniformly at random and independently of ρ.

2. *With probability at least $1-\varepsilon$ over ρ, the restricted function $B|_\rho\colon \{0,1\}^n \to \{0,1\}$ can be computed by a width-3 standard-order ROBP $\overline{B}$ with the property that for every vertex v, we have*

$$\mathsf{Weight}(\overline{B}^{v\leftarrow}) \le \mathrm{polylog}(n/\varepsilon).$$

Before proving Theorem 7.2, we first show that the combination of Theorem 7.2 and Theorem 6.3 implies Theorem 1.5, which is restated next.

Theorem 7.3 (Theorem 1.5, restated). *For every $n \in \mathbb{N}$ and $\delta > 0$, there is an explicit WPRG with seed length*

$$\widetilde{O}\left(\log n \cdot \sqrt{\log(1/\delta)} + \log(1/\delta)\right)$$

that δ-fools width-3 length-n standard-order ROBPs.

Proof, assuming Theorem 7.2. Let $(\mathcal{G}, \mu)$ be the WPRG from Theorem 6.3 with a parameter W to be specified later and with error $\delta/3$. Theorem 6.3 guarantees that this WPRG is K-bounded for some value $K = n^{O(\sqrt{\log(1/\delta)})}$. Let $\mathcal{R}$ be the restriction generator from Theorem 7.2 with $\varepsilon = \frac{\delta}{3\cdot(K+1)}$. Our WPRG $(\mathcal{G}', \mu')$ applies $\mathcal{R}$ and then applies $(\mathcal{G}, \mu)$. That is, we define $\mathcal{G}'(x,y) = \mathcal{G}(y) \circ \mathcal{R}(x)$ and $\mu'(x,y) = \mu(y)$. Let B be a width-3 length-n standard-order ROBP. Our definition ensures that

$$\mathbb{E}_{x,y}[B(\mathcal{G}'(x,y)) \cdot \mu'(x,y)] = \mathbb{E}_{x,y}[B|_{\mathcal{R}(x)}(\mathcal{G}(y)) \cdot \mu(y)].$$

Let $\rho = \mathcal{R}(x)$ for a uniform random x. We consider the contribution to the error from two cases:

[12] Here, we only define $B|_\rho$ as a *function*. Later, we will refine the definition by specifying a particular *standard-order ROBP* computing $B|_\rho$, in the case that B is itself a standard-order ROBP. See Section 7.1.

- *Case 1: All subprograms of the restricted program have low weight.* Theorem 7.2 ensures that with probability $1-\varepsilon$ over ρ, the function $B|_\rho$ can be computed by a program $\overline{B}$ such that for every vertex v, $\mathsf{Weight}(\overline{B}^{v\leftarrow}) \le W$ where $W = \mathrm{polylog}(n/\varepsilon)$ (this is the value W that we use when invoking Theorem 6.3). In such a case, Theorem 6.3 guarantees that $(\mathcal{G}, \mu)$ fools $B|_\rho$ with error at most $\delta/3$.

- *Case 2: Not all subprograms of the restricted program have low weight.* With probability at most ε, we have no low-weight guarantee. In such a case, the error from the WPRG $(\mathcal{G}, \mu)$ could be larger than 1! However, the error is at most $K+1$, because

$$|\mathbb{E}_y[B|_\rho(\mathcal{G}(y)) \cdot \mu(y)] - \mathbb{E}[B_\rho]| \le \max_y |B|_\rho(\mathcal{G}(y))| \cdot |\mu(y)| + \mathbb{E}[B|_\rho] \le K+1.$$

  Therefore, the contribution to the error from this case is at most $\varepsilon \cdot (K+1) = \delta/3$.

Since $|\mathbb{E}[B] - \mathbb{E}_{\rho,\mathcal{U}}[B|_\rho(\mathcal{U})]| \le \varepsilon < \delta/3$ and since $(\mathcal{G}, \mu)$ adds at most $\delta/3 + \delta/3$ error, the overall error is at most δ.

The seed length for the restriction generator $\mathcal{R}$ is

$$\widetilde{O}(\log(n/\varepsilon)) = \widetilde{O}(\log(Kn/\delta)) = \widetilde{O}\left(\log n \cdot \sqrt{\log(1/\delta)} + \log(1/\delta)\right).$$

The seed length for the WPRG $(\mathcal{G}, \mu)$ is

$$\widetilde{O}\left(\log n \cdot \left(\log(W \cdot 3) + \sqrt{\log(1/\delta)}\right) + \log(1/\delta)\right) = \widetilde{O}\left(\log n \cdot \sqrt{\log(1/\delta)} + \log(1/\delta)\right),$$

because $\log W = O(\log\log(n/\varepsilon)) = O(\log\log(n/\delta))$. □

The rest of this section is devoted to proving Theorem 7.2.

7.1 Syntactic Restrictions

To prove Theorem 7.2, we will heavily rely on the work of Forbes and Kelley [FK18]. They showed how to sample pseudorandom restrictions that assign values to a constant fraction of the variables, while preserving the expectation of any constant-width ROBP up to error ε, using a seed length of $\widetilde{O}(\log(n/\varepsilon))$. Given their work, the nontrivial part of the proof of Theorem 7.2 is establishing the second conclusion: we will show that after $O(\log\log(n/\varepsilon))$ rounds of Forbes-Kelley restrictions, with high probability, all subprograms $\overline{B}^{v\leftarrow}$ have low weight, where $\overline{B} \equiv B|_\rho$.

Roughly speaking, our approach will be to first show that $B|_\rho$ itself has low weight with high probability, and then take a union bound over all target vertices v. Let us highlight one subtle aspect of this "union bound" argument. Essentially, the argument will show that $(B^{v\leftarrow})|_\rho$ (the restriction of a subprogram) has low weight. However, we would like to bound the weight of $(B|_\rho)^{v\leftarrow}$ (a subprogram of the restricted program). Could the two be different?

We carefully set up our definitions to ensure that $(B^{v\leftarrow})|_\rho$ and $(B|_\rho)^{v\leftarrow}$ are the same program. The key is to work with "syntactic" restrictions rather than "semantic" restrictions. That is, in the remainder of this section, we will be relatively fastidious about distinguishing between an ROBP and the function it computes; we will think of restrictions as acting on ROBPs rather than on functions. The restriction of an ROBP is another ROBP defined as follows.

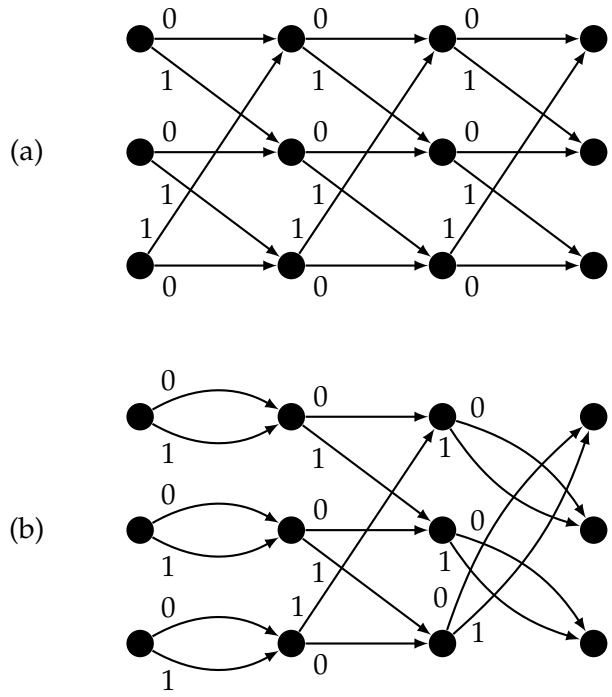

Figure 4: An example of (a) a width-3 ROBP B and (b) the syntactically-restricted program $B|_\rho$ where $\rho = (0, \star, 1)$.

Definition 7.4 (Syntactic restriction of an ROBP)**.** *Let B be a width-w length-n standard-order ROBP with vertex set $V(B) = V^{(0)} \cup \cdots \cup V^{(n)}$. Let $\rho \in \{0,1,\star\}^n$ be a restriction. The* syntactically-restricted program *$B|_\rho$ is a width-w length-n ROBP on the same vertex set, $V(B|_\rho) = V^{(0)} \cup \cdots \cup V^{(n)}$, with edges defined as follows. Let $i \in [n]$.*

- *If $\rho_i = \star$, then each vertex $u \in V^{(i-1)}$ has the same outgoing edges in $B|_\rho$ as it does in B.*
- *If $\rho_i = b \in \{0,1\}$, then for each vertex $u \in V^{(i-1)}$, both outgoing edges in $B|_\rho$ lead to the same vertex $v \in V^{(i)}$, namely, the vertex v such that (u, v) is an edge in B with label b.*

No additional simplifications are performed. (See Figure 4.)

As suggested previously, Definition 7.4 satisfies the following two basic properties:

Fact 7.5 (Syntactic restrictions are compatible with semantic restrictions)**.** *Let B be a length-n standard-order ROBP, let $\rho \in \{0,1,\star\}^n$, and let $x \in \{0,1\}^n$. Then $B|_\rho(x) = B(x \circ \rho)$.*

Fact 7.5 shows that Definition 7.4 is compatible with (and refines) the standard "semantic" definition of the restriction of a function (Definition 7.1).

Fact 7.6 (Subprogram of restriction equals restriction of subprogram)**.** *Let B be a standard-order ROBP, let $\rho \in \{0,1,\star\}^n$, and let v be a vertex. Then $(B^{v\leftarrow})|_\rho$ and $(B|_\rho)^{v\leftarrow}$ are precisely the same program.*

Remark 7.7. *The conclusions of Theorem 7.2 only say something about the* function computed by *$B|_\rho$. Therefore, when reading and using the* statement *of Theorem 7.2, the expression $B|_\rho$ can safely be interpreted as a semantic restriction or a syntactic restriction. The statement is equivalent either way. It is only in the* proof *of Theorem 7.2 that we rely on the notion of a syntactic restriction.*

7.2 Deleting Unreachable Vertices

Working with syntactic restrictions allows us to ignore the issue of subprograms and focus on bounding the weight of $B|_\rho$. Unfortunately, it is not possible to literally show $\mathsf{Weight}(B|_\rho) = \mathrm{polylog}(n/\varepsilon)$ (see Figure 5). Instead, we bound the weight of the closely related program $\mathsf{DelUnreach}(B|_\rho)$, defined below.

Definition 7.8 (Reachability)**.** *Let B be a standard-order ROBP. We say that a vertex v is* reachable *if there is a path from the start vertex to v. Otherwise, we say that v is* unreachable*.*

Definition 7.9 (Deleting unreachable vertices)**.** *Let B be a standard-order ROBP. We define $\mathsf{DelUnreach}(B)$ by deleting all unreachable vertices along with their outgoing edges.*[13]

We will prove the following.

Theorem 7.10 (Syntactic pseudorandom restrictions for width-3 ROBPs)**.** *For every $n \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit restriction generator $\mathcal{R}\colon \{0,1\}^s \to \{0,1,\star\}^n$ with seed length $s = \widetilde{O}(\log(n/\varepsilon))$ such that for every width-3 standard-order ROBP B, if we sample $\rho = \mathcal{R}(U_s)$, then*

1. *The restriction ρ* preserves the expectation *of B up to error ε. That is,*

$$|\mathbb{E}_{\rho,\mathcal{U}}[B|_\rho(\mathcal{U})] - \mathbb{E}[B]| \le \varepsilon,$$

   where $\mathcal{U}$ is sampled uniformly at random and independently of ρ.

2. *With probability at least $1-\varepsilon$ over ρ, we have $\mathsf{Weight}(\mathsf{DelUnreach}(B|_\rho)) \le \mathrm{polylog}(n/\varepsilon)$, where $B|_\rho$ is the* syntactic *restriction of B (see Definition 7.4).*

Remark 7.11. *In general, $\mathsf{Weight}(\mathsf{DelUnreach}(B))$ depends on both the start vertex of B and the accepting vertices. In contrast, $\mathsf{Weight}(B)$ does not depend on the start vertex of B, although it does depend on the accepting vertices. The syntactically-restricted program $B|_\rho$ has the same start vertex and the same accepting vertices as the original program B.*

Before proving Theorem 7.10, let us see how Theorem 7.2 will follow from it.

Proof of Theorem 7.2, assuming Theorem 7.10. Apply Theorem 7.10 with error $\varepsilon/(3n)$, and let $\overline{B} = \mathsf{DelUnreach}(B|_\rho)$.[14] By the union bound, with probability $1-\varepsilon$, for every vertex v, we have

$$\mathsf{Weight}(\mathsf{DelUnreach}(B^{v\leftarrow}|_\rho)) \le \mathrm{polylog}(n/\varepsilon).$$

Assume that this occurs. Because of our syntactic notion of restriction (Definition 7.4), the programs $(B^{v\leftarrow})|_\rho$ and $(B|_\rho)^{v\leftarrow}$ are identical (Fact 7.6). Furthermore, for every vertex v in $\overline{B}$, the definition of DelUnreach implies that the programs $\mathsf{DelUnreach}((B|_\rho)^{v\leftarrow})$ and $\mathsf{DelUnreach}(B|_\rho)^{v\leftarrow}$ are identical. Therefore,

$$\mathsf{Weight}(\overline{B}^{v\leftarrow}) = \mathsf{Weight}(\mathsf{DelUnreach}((B|_\rho)^{v\leftarrow})) = \mathsf{Weight}(\mathsf{DelUnreach}(B^{v\leftarrow}|_\rho)) \le \mathrm{polylog}(n/\varepsilon). \quad \square$$

The remainder of Section 7 consists of the proof of Theorem 7.10.

[13] Note that in this section, we consider programs where the width of a layer, $|V^{(i)}|$, might vary from one layer to the next. In contrast, in the rest of the paper, we assume without loss of generality that $|V^{(i)}| = w$ for every i.

[14] The reader might be concerned by the fact that in $\overline{B}$, different layers have different widths, whereas in Section 6, we assume that every layer has width precisely w. The two models are equivalent, because we can add dummy vertices. As long as the two outgoing edges of each dummy vertex point to the same vertex in the next layer, these new dummy edges have zero weight. Equivalently, to define $\overline{B}$, instead of deleting the unreachable vertices in $B|_\rho$, we can redirect the outgoing 1-edge of each unreachable vertex to go to the same vertex that the 0-edge goes to.

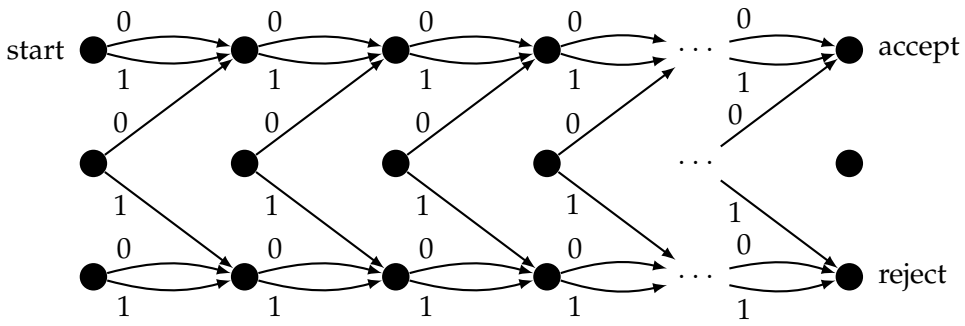

Figure 5: An example showing the necessity of deleting unreachable vertices. For the branching program B depicted above, we have $\mathsf{Weight}(B|_\rho) = |\rho^{-1}(\star)|$ for every restriction ρ. In particular, in the "mild restriction" regime $|\rho^{-1}(\star)| = n^{1-o(1)}$, the syntactically-restricted program has high weight – but only because of unreachable edges.

7.3 Restrictions Eliminate Colliding Layers

Following Meka, Reingold, and Tal [MRT19], our proof of Theorem 7.10 centers on the concept of a *colliding layer*.

Definition 7.12 (Colliding layer)**.** *Let B be a standard-order ROBP on vertex set $V = V^{(0)} \cup \cdots \cup V^{(n)}$ and let $i \in [n]$. We say that layer i is* colliding *if there are two distinct edges that have the same label and that lead to the same vertex $v \in V^{(i)}$.*

A program with no colliding layers (a permutation program) is regular, and hence it has bounded weight by the results of Braverman, Rao, Raz, and Yehudayoff [BRRY14]. Roughly speaking, our plan is to argue that pseudorandom restrictions reduce the number of colliding layers in the program (similar to the arguments of Meka, Reingold, and Tal [MRT19]). We will mainly be focused on counting the number of colliding layers *between width-2 layers*. (Again, this is similar to the arguments of Meka, Reingold, and Tal [MRT19]). More precisely, we use the following definition, which refines Meka, Reingold, and Tal's notion of a (w, ℓ, m)-ROBP.

Definition 7.13 ((w, i, ℓ, m, r) program)**.** *Let B be a width-w length-n standard-order ROBP, and let $i, \ell, m, r \in \mathbb{N}$. We say that B is a (w, i, ℓ, m, r) program if there are indices $i = i_0 < i_1 < \cdots < i_k = n$ such that the following hold.*

- *Layer i_j has width at most 2 for each $j \in \{0, \ldots, k\}$.*
- *The intervals $(i_0, i_1], \ldots, (i_{k-1}, i_k]$ can be partitioned into two categories, "ordinary" and "unruly," such that the following hold.*
  - *There are at most m ordinary intervals, and there are at most ℓ colliding layers in each ordinary interval.*
  - *There are at most r colliding layers in total among all the unruly[15] intervals.*

Note that Definition 7.13 says nothing about layers $1, 2, \ldots, i-1$. Indeed, later we will treat layers $i, i+1, \ldots, n$ as an "approximation" for the entire program, similar to Meka, Reingold, and Tal [MRT19].

[15]The term "unruly" is borrowed from a similar concept in work by Doron, Meka, Reingold, Tal, and Vadhan [DMRTV21].

7.3.1 Initial Restriction: Bringing ℓ Down to $O(\log n)$

Following Meka, Reingold, and Tal [MRT19], we view the parameter "ℓ" as our primary measure of the complexity of the program. Initially, we can trivially take $\ell = n$. Meka, Reingold, and Tal observed [MRT19] that it is fairly straightforward to design a pseudorandom restriction that brings the parameter ℓ down to $O(\log n)$. For example, it suffices for the restriction to be *k-wise δ-close to a truly random restriction*, as defined below.

Definition 7.14 (Truly random restriction)**.** *Let $\mathcal{R}_p$ denote the distribution over $\rho \in \{0,1,\star\}^p$ in which the coordinates are independent and*

$$\rho_i = \begin{cases} \star & \text{with probability } p \\ 0 & \text{with probability } (1-p)/2 \\ 1 & \text{with probability } (1-p)/2. \end{cases}$$

Definition 7.15 (k-wise δ-close)**.** *Let ρ, ρ' be two random variables distributed over $\{0,1,\star\}^n$. We say that ρ is* k-wise δ-close *to ρ' if, for every set $S \subseteq [n]$ of size at most k, the substrings ρ_S and ρ'_S are δ-close in total variation distance.*

To convert some width-3 layers into width-2 layers, we will apply a pseudorandom restriction and then delete unreachable vertices:

Claim 7.16 ($\ell \to O(\log n)$)**.** *Let B be a standard-order width-3 length-n ROBP, let $p \in (0, 1/2]$, and let $\delta > 0$. Assume that the first and last layers of B have width at most 2. There is a value $\ell = O(\log(n/\delta))$ such that if $\rho \in \{0,1,\star\}^n$ is a random variable that is ℓ-wise $(\frac{\delta}{2n})$-close to $\mathcal{R}_p$, then with probability at least $1-\delta$, the program $\mathsf{DelUnreach}(B|_\rho)$ is a $(3, 0, \ell, n, 0)$ program.*

Proof. Let $\mathcal{I} \subseteq [n]$ be the set of colliding layers of B. For each $i \in \mathcal{I}$, there is some $b_i \in \{0,1\}$ such that there are two edges in $V^{(i-1)} \times V^{(i)}$ that are both labeled b_i and that point to the same vertex. Observe that if $\rho_i = b_i$ for some $i \in \mathcal{I}$, then layer i has width at most 2 in $\mathsf{DelUnreach}(B|_\rho)$.

We say that $i \in \mathcal{I}$ is *bad* if there is some j such that $|\mathcal{I} \cap [i,j]| \geq \ell$, and for every $i' \in \mathcal{I} \cap [i,j]$, we have $\rho_{i'} \neq b_{i'}$. The probability that i is bad is at most $(1/2 + p/2)^\ell + \delta/(2n)$, which is at most δ/n for a suitable choice of $\ell = O(\log(n/\delta))$. By the union bound, we may assume that there is no bad i. In this case, $\mathsf{DelUnreach}(B|_\rho)$ is a $(3, 0, \ell, n, 0)$ program, because if some layer i is colliding in $\mathsf{DelUnreach}(B|_\rho)$, then layer i was already colliding in B, i.e., $i \in \mathcal{I}$. □

Claim 7.16 shows that after applying a pseudorandom restriction and deleting unreachable vertices, with high probability, we have $\ell = O(\log n)$. However, we will need to show something stronger, which is that ℓ is likely to be low after we apply a pseudorandom restriction, delete unreachable vertices, *and then merge equivalent vertices*.

Definition 7.17 (Locally equivalent vertices [MRT19] and equivalent vertices)**.** *Let B be a standard-order ROBP with layers $V^{(0)}, \ldots, V^{(n)}$. Let $u, v \in V^{(i)}$ where $i \in \{0, \ldots, n-1\}$. We say that u and v are* locally equivalent *if $B[u,0] = B[v,0]$ and $B[u,1] = B[v,1]$.*[16]

Now let $u, v \in V^{(i)}$ where $i \in \{0, \ldots, n\}$. We say that u and v are equivalent *if for every x, we have $B^{\leftarrow u}(x) = B^{\leftarrow v}(x)$.*[17] *We say that B has* no duplicate vertices *if there are no two distinct vertices that are equivalent.*

Observe that any locally-equivalent vertices are also equivalent.

[16] Recall that $B[u,x]$ is the vertex reached from u by reading the string x.

[17] Recall that $B^{\leftarrow u}(x)$ is the Boolean output value of the program given input x from start vertex u.

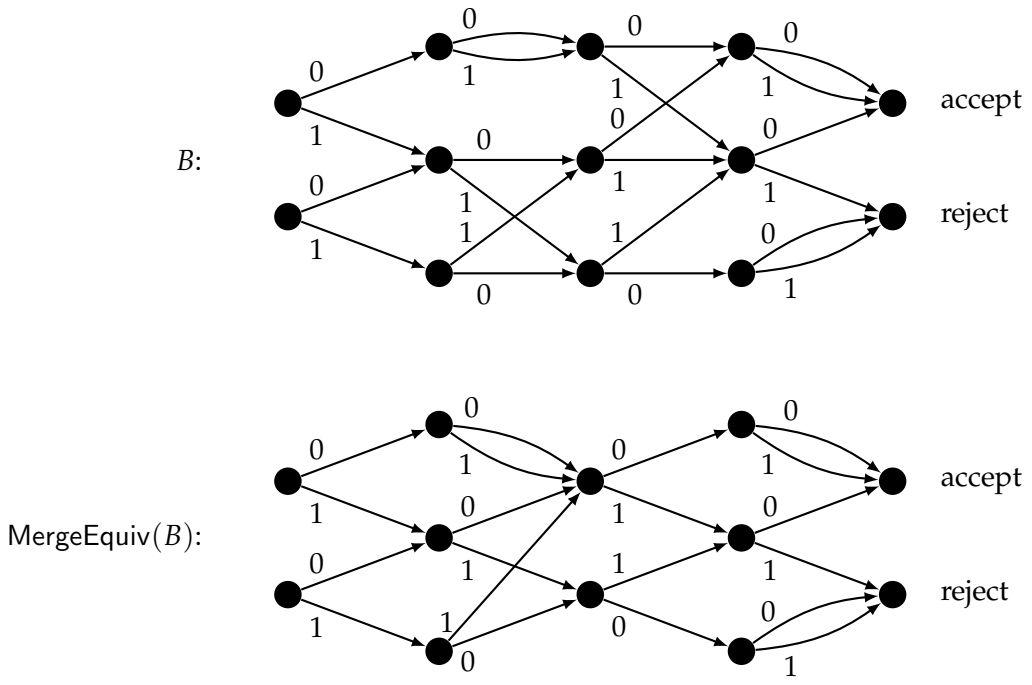

Figure 6: An example showing that MergeEquiv$(\cdot)$ can create new colliding layers. In this example, B has two colliding layers, but MergeEquiv(B) has three colliding layers.

Definition 7.18 (The MergeEquiv operation)**.** *Let B be a length-n standard-order ROBP. We define* MergeEquiv(B)*, another length-n standard-order ROBP, by merging equivalent vertices until we get a program with no duplicate vertices.*

We emphasize that the MergeEquiv$(\cdot)$ operation merges equivalent vertices within a single layer, but it never contracts layers. The number of layers in MergeEquiv(B) is the same as the number of layers in B.

The MergeEquiv$(\cdot)$ operation can sometimes introduce new colliding layers (see Figure 6), but we'll show that it doesn't increase the parameter ℓ. The following claim will help us to reason about these newly-created colliding layers. Essentially, the claim says that if layer i is a newly-created colliding layer, then some vertices in layer i got merged, but no vertices in layer $i-1$ got merged.

Claim 7.19 (Describing newly-colliding layers)**.** *Let B be a width-3 standard-order ROBP. Suppose that layer i is not colliding in B, but it is colliding in* MergeEquiv(B)*. Then the width of layer i is smaller in* MergeEquiv(B) *than it is in B, whereas the width of layer $i-1$ is the same in* MergeEquiv(B) *as it is in B.*

Proof. Since layer i is colliding in MergeEquiv(B), there are two edges $e = (u, v)$ and $e' = (u', v)$ in the layer, both with the same label $b \in \{0, 1\}$, where $u \neq u'$. Since layer i was not colliding in B, the vertex v must be the result of merging two vertices v', v'' in layer i of B. This shows that the width of layer i is smaller in MergeEquiv(B) than it is in B.

Now let us analyze layer $i-1$. Since $u \neq u'$, layer $i-1$ has width at least two in MergeEquiv(B). Suppose there was a third vertex in layer $i-1$ of B, say u''. The b-edge leaving u'' in B must not have pointed at v' or v'', because layer i is not colliding in B. Therefore, the b-edge leaving u'' must have pointed at a third vertex, v'''. This vertex v''' cannot have merged with v' and v'', because if it had, then layer i would be width-1 in MergeEquiv(B), implying that all previous layers are

width-1, contradicting the fact that $u \neq u'$. Therefore, v''' is still present in $\mathsf{MergeEquiv}(B)$, and hence u'' was not merged with either of the other vertices in layer $i-1$. Thus, the width of layer $i-1$ is the same in $\mathsf{MergeEquiv}(B)$ as it is in B. □

Using Claim 7.19, we can show that the conclusion of Claim 7.16 remains true even after applying the $\mathsf{MergeEquiv}(\cdot)$ operator.

Claim 7.20. *Let B be a $(3, 0, \ell, m, 0)$ program. Then $\mathsf{MergeEquiv}(B)$ is a $(3, 0, \ell, m', 0)$ program for some m'.*

Proof. Let $i < j$, and suppose that in $\mathsf{MergeEquiv}(B)$, layers i and j have width at most two, whereas layers $i+1, i+2, \ldots, j-1$ all have width 3. Let $j' > i$ be the first layer that has width at most two in B. Since the $\mathsf{MergeEquiv}(\cdot)$ operator can never increase the width of a layer, we must have $j' \geq j$.

Since B is a $(3, 0, \ell, m, 0)$ program, at most ℓ of the layers in the interval $(i, j']$ are colliding in B. By Claim 7.19, every layer in the interval (i, j) that is colliding in $\mathsf{MergeEquiv}(B)$ must have already been colliding in B. Furthermore, layer j' must have been colliding in B, simply because layer j' has width at most two and layer $j'-1$ has width three.[18] Consequently, at most $\ell - 1$ of the layers in the interval (i, j) are colliding in $\mathsf{MergeEquiv}(B)$, and therefore at most ℓ of the layers in the interval $(i, j]$ are colliding in $\mathsf{MergeEquiv}(B)$. □

7.3.2 Subsequent Restrictions: Bringing ℓ Down to $\ell/2$

Claim 7.16 and Claim 7.20 show that after applying *one* pseudorandom restriction and the $\mathsf{DelUnreach}(\cdot)$ and $\mathsf{MergeEquiv}(\cdot)$ operators, with high probability, we have $\ell = O(\log n)$. Now we would like to argue that applying additional pseudorandom restrictions decreases ℓ even further, even after we merge equivalent vertices.

After the first couple of rounds of restrictions, we will have a $(3, i, \ell, m, r)$ program where $r > 0$. We must therefore analyze the effect of $\mathsf{MergeEquiv}(\cdot)$ on unruly intervals. The following claim will help us to do so.

Claim 7.21 (Merging $\implies$ locally equivalent vertices)**.** *Let B be a standard-order ROBP and let $i \in \{0, \ldots, n-1\}$. Suppose that the width of layer i is smaller in $\mathsf{MergeEquiv}(B)$ than it is in B, but the width of layer $i+1$ is the same in $\mathsf{MergeEquiv}(B)$ as it is in B. Then there are two distinct locally equivalent vertices in layer i of B.*

Proof. Let $V^{(0)}, \ldots, V^{(n)}$ be the layers of B. Since the width of layer i is smaller in $\mathsf{MergeEquiv}(B)$ than it is in B, there are two distinct but equivalent vertices $u, v \in V^{(i)}$. Let $b \in \{0, 1\}$, and let (u, u') and (v, v') be the outgoing edges of u and v with label b. Since u and v are equivalent, u' and v' must be equivalent. Since the width of layer $i+1$ is the same in $\mathsf{MergeEquiv}(B)$ as it is in B, layer $i+1$ does not contain any pair of distinct, equivalent vertices. Therefore, $u' = v'$. Consequently, u and v are *locally* equivalent. □

We will use the following variant of Claim 7.19.

Claim 7.22. *Let B be a width-3 standard-order ROBP, and let ρ be a restriction. Suppose that layer i is not colliding in B, but it is colliding in $\mathsf{MergeEquiv}(B|_\rho)$. Then the width of layer i is smaller in $\mathsf{MergeEquiv}(B|_\rho)$ than it is in B, whereas the width of layer $i-1$ is the same in $\mathsf{MergeEquiv}(B|_\rho)$ as it is in B.*

[18] An edge case is when $j' = j = i+1$, so we cannot say that layer $j'-1$ has width three. To handle this case, if $\ell > 0$, we can observe that the interval $(i, j]$ only contains a single interval. On the other hand, if $\ell = 0$, then B does not have any colliding layers, and its final layer has at most two vertices, so B and $\mathsf{MergeEquiv}(B)$ are the same program.

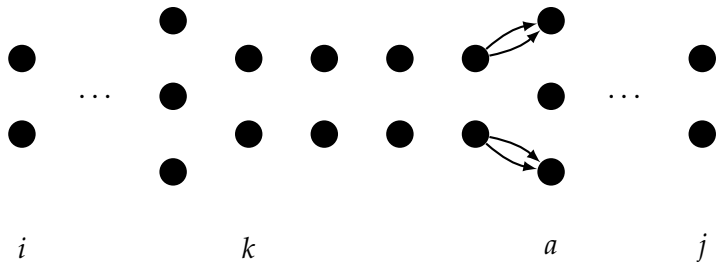

Figure 7: The program $\mathsf{MergeEquiv}(B|_\rho)$ in the proof of Claim 7.23. For each newly-created colliding layer (k), we identify a corresponding layer (a) that was colliding in B but that is no longer colliding in $\mathsf{MergeEquiv}(B|_\rho)$.

Proof. Since layer i is not colliding in B, it is also not colliding in $B|_\rho$. Therefore, by Claim 7.19, the width of layer i is smaller in $\mathsf{MergeEquiv}(B|_\rho)$ than it is in $B|_\rho$, whereas the width of layer $i-1$ is the same in $\mathsf{MergeEquiv}(B|_\rho)$ as it is in $B|_\rho$. Finally, each layer has the same width in $B|_\rho$ as it does in B. □

Using Claim 7.22 and Claim 7.21, we can prove the following claim, which says that between any two width-2 layers, the total number of colliding layers can never increase when we apply a restriction and then apply the $\mathsf{MergeEquiv}(\cdot)$ operation. Think of $\mathcal{I}$ as an unruly interval.

Claim 7.23. *Let B be a width-3 standard-order ROBP with no duplicate vertices, and let ρ be a restriction. Let $\mathcal{I} = (i, j]$ where $i < j$. Suppose that layers i and j have width at most two in B, and suppose that q of the layers in $\mathcal{I}$ are colliding in B. Then at most q of the layers in $\mathcal{I}$ are colliding in $\mathsf{MergeEquiv}(B|_\rho)$.*

Proof. Let $\mathcal{S}$ be the set of layers in $\mathcal{I}$ that are colliding in B, and let $\mathcal{S}'$ be the set of layers in $\mathcal{I}$ that are colliding in $\mathsf{MergeEquiv}(B|_\rho)$. We will define an injective function $\phi\colon \mathcal{S}' \to \mathcal{S}$. The existence of such a function will imply that $|\mathcal{S}'| \le |\mathcal{S}| = q$.

Fix $k \in \mathcal{S}'$. If $k \in \mathcal{S}$, then we define $\phi(k) = k$. Assume now that $k \notin \mathcal{S}$, i.e., k is a "newly-created" colliding layer. By Claim 7.22, the width of layer k is smaller in $\mathsf{MergeEquiv}(B|_\rho)$ than it is in B. In contrast, observe that the width of layer j must be the same in $\mathsf{MergeEquiv}(B|_\rho)$ as it is in B, because it already has width at most 2 in B, and if it had width 1 in $\mathsf{MergeEquiv}(B|_\rho)$, then all previous layers would also have width 1, contradicting the fact that layer k is a colliding layer in $\mathsf{MergeEquiv}(B|_\rho)$. Therefore, $j > k$, and if we let a be the first layer after k that has the same width in $\mathsf{MergeEquiv}(B|_\rho)$ as it has in B, then $a \in \mathcal{I}$. By Claim 7.21, there are two distinct vertices u, v in layer $a-1$ of $B|_\rho$ that are locally equivalent. Since B has no duplicate vertices, u and v must not be locally equivalent in B. Therefore, $\rho_a \in \{0,1\}$ and layer a is colliding in B, i.e., $a \in \mathcal{S}$. Let $\phi(k) = a$, and note that $\phi(k) > k$. (See Figure 7.)

To show that ϕ is injective, let $k' \in \mathcal{S}'$ with $k' \neq k$. We will show that $\phi(k) \neq \phi(k')$. First, suppose $k' \in \mathcal{S}$, so $\phi(k') = k' \in \mathcal{S}'$. Observe that $a \notin \mathcal{S}'$, because $\rho_a \in \{0,1\}$. Therefore, $\phi(k) \neq \phi(k')$ in this case.

Now, suppose instead that $k' \notin \mathcal{S}$, i.e., k' is a "newly-created" colliding layer (like k). By swapping the roles of k and k' if necessary, we may assume without loss of generality that $k' > k$. Recall that a is the first layer after k that has the same width in $\mathsf{MergeEquiv}(B|_\rho)$ as it has in B. Therefore, by Claim 7.22, there are no newly-created colliding layers in the interval $(k, a]$. Therefore, $\phi(k') > k' > a = \phi(k)$, so $\phi(k') \neq \phi(k)$. □

Using Claim 7.23, we now show that if a restriction assigns values to enough variables, then the parameter ℓ decreases, while r only increases by a modest amount.

Claim 7.24 (Restrictions eliminate colliding layers)**.** *Let B be a $(3, i, \ell, m, r)$ program with no duplicate vertices. Let ρ be a restriction such that in all but k of the ordinary intervals, ρ assigns values to all but at most ℓ' colliding layers. Then* $\mathsf{MergeEquiv}(B|_\rho)$ *is a $(3, i, \ell' + 1, m', r + \ell k)$ program for some m'.*

Proof. Let $i \leq j < j'$, suppose that layers j and j' have width at most two in $\mathsf{MergeEquiv}(B|_\rho)$, and suppose that layers $j+1, j+2, \ldots, j'-1$ all have width three in $\mathsf{MergeEquiv}(B|_\rho)$. Since the $\mathsf{MergeEquiv}(\cdot)$ operator can never increase the width of a layer, the interval $(j, j']$ must be entirely contained within either an ordinary interval of B or an unruly interval of B.

Suppose first that $(j, j']$ is contained in an ordinary interval $\mathcal{I}$ of B, and furthermore, suppose that ρ assigns values to all but at most ℓ' of the layers in $\mathcal{I}$ that are colliding in B. In this case, we will consider $(j, j']$ to be an ordinary interval of $\mathsf{MergeEquiv}(B|_\rho)$. Whenever ρ assigns a value to some layer, that layer cannot be colliding in $\mathsf{MergeEquiv}(B|_\rho)$. Furthermore, by Claim 7.22, any layer in the interval (j, j') that is colliding in $\mathsf{MergeEquiv}(B|_\rho)$ must have already been colliding in B. Therefore, at most ℓ' of the layers in the interval (j, j') can be colliding in $\mathsf{MergeEquiv}(B|_\rho)$. Including layer j', at most $\ell' + 1$ of the layers in the interval $(j, j']$ can be colliding in $\mathsf{MergeEquiv}(B|_\rho)$.

Otherwise, we designate $(j, j']$ as an unruly interval of $\mathsf{MergeEquiv}(B|_\rho)$. Claim 7.23 implies that among all these unruly intervals, the total number of colliding layers in $\mathsf{MergeEquiv}(B|_\rho)$ is indeed at most $r + \ell k$. □

Claim 7.24 implies that if a distribution over restrictions ρ has a certain mild "pseudorandomness" property, and if we initially start with $m \leq \exp(O(\ell))$, then with high probability, after we apply ρ and we apply the $\mathsf{MergeEquiv}(\cdot)$ operation, the parameter ℓ will decrease by a factor of two:

Claim 7.25 ($\ell \to \ell/2$)**.** *For each constant $C \in \mathbb{N}$, there is a constant p such that the following holds. Let B be a $(3, i, \ell, C^\ell, r)$ program with no duplicate vertices where $\ell \geq 4$. Let ρ be a random variable distributed over $\{0, 1, \star\}^n$ with the property that for every set $S \subseteq [n]$ with $|S| \leq \log(1/\delta)$, we have*

$$\Pr[\rho_S = \star^S] \leq p^{|S|} + \delta.$$

Then with probability at least $1 - \sqrt{\delta}$, the program $\mathsf{MergeEquiv}(B|_\rho)$ *is a $(3, i, \ell/2, m', r + \log(1/\delta))$ program for some m'.*

Note that the assumption on ρ is implied by the condition of being $\log(1/\delta)$-wise δ-close to $\mathcal{R}_p$.

Proof. Let $k = \log_{2C}(0.5/\sqrt{\delta})/\ell$. For any set of k ordinary intervals, the probability that ρ leaves at least $\ell/2 - 1$ colliding layers alive in each interval is at most

$$2^{\ell k} \cdot (p^{(\ell/2-1)\cdot k} + \delta) \leq 2^{\ell k} \cdot (p^{\ell k/4} + \delta) \leq 2^{\ell k} \cdot 2\delta,$$

provided we choose $p = p(C)$ to be a small enough constant. Therefore, the probability that there exist k ordinary intervals in which ρ leaves at least $\ell/2 - 1$ colliding layers alive is at most $(C^\ell)^k \cdot 2^{\ell k} \cdot 2\delta = (2C)^{\ell k} \cdot 2\delta = \sqrt{\delta}$. If this bad event does not occur, then by Claim 7.24, the program $\mathsf{MergeEquiv}(B|_\rho)$ is a $(3, i, \ell/2, m', r + \ell k)$ program for some m'. □

7.4 Approximation by Suffix Programs

We would like to apply Claim 7.25 repeatedly, so we can continue to decrease ℓ. However, Claim 7.25 is only applicable when $m \leq \exp(O(\ell))$. To ensure that this assumption holds, similar to Meka, Reingold, and Tal [MRT19], we will approximate an ROBP using one of its "suffixes," i.e., layers i through n for some $i > 0$. We measure the error of the approximation using the following definition.

Definition 7.26 (Error when approximating a program using a suffix)**.** *Let B be a standard-order ROBP with vertex set $V(B) = V^{(0)} \cup \cdots \cup V^{(n)}$. Let $i \in \{0, \ldots, n\}$. We define*

$$\mathsf{Err}(B, i) = \Pr_{x \in \{0,1\}^n}[\text{there exist } u, v \in V^{(i)} \text{ such that } B^{\leftarrow u}(x) \neq B^{\leftarrow v}(x)].$$

Note that if the bad event above does not occur, then $B(x) = B^{\leftarrow v}(x)$ for every $v \in V^{(i)}$, and thus the first i steps of the computation have no effect on the final outcome. The most important case to keep in mind is when layer i has width 2, say $V^{(i)} = \{u, v\}$, so $\mathsf{Err}(B, i) = \Pr_x[B^{\leftarrow u}(x) \neq B^{\leftarrow v}(x)]$. This case is ultimately the only one that "matters" in our analysis, but we use the more general definition of $\mathsf{Err}(B, i)$ for ease of exposition.

Remark 7.27. *In Meka, Reingold, and Tal's analysis [MRT19], they look at the suffix (layers $i, i+1, \ldots, n$) as a program of length $n - i$, and they think of it as having "multiple start vertices" in a certain sense. The intuition behind our argument is the same, but we use a different formalism, as stipulated by Definition 7.26.*

Consider a $(3, 0, \ell, m, r)$ program B. We wish to argue that if layers $i, \ldots, n$ include $\exp(\Theta(\ell))$ ordinary intervals, then $\mathsf{Err}(B, i)$ is low. This will allow us to ignore the layers before layer i and thus effectively ensure $m = \exp(\Theta(\ell))$. To prove it, the idea is that if an interval starts with a width-two layer $\{u, v\}$ and it has a colliding layer, then that colliding layer is an "opportunity" for the two paths from u and v to collide, which would imply that $B^{\leftarrow u}(x) = B^{\leftarrow v}(x)$. For this argument to make sense, we need to ensure that the collision can actually be realized. To do so, we wish to understand the colliding layers that remain *after deleting unreachable vertices* from the program. Under the assumption that there are neither unreachable vertices nor duplicate vertices (see Definition 7.8 and Definition 7.17), Meka, Reingold, and Tal showed [MRT19] that indeed, there is a noticeable chance that the two paths from u and v collide:

Claim 7.28 ([MRT19, Claims 7.5, 7.6, and 7.7])**.** *Let B be a width-3 ROBP with no unreachable vertices and no duplicate vertices. Let $i < j$, and suppose that layers i and j have width at most 2. Let $V^{(i)} = \{u, v\}$. Suppose that the number of colliding layers in the interval $(i, j]$ is at least one and at most ℓ. Pick $\mathcal{U} \in \{0,1\}^{j-i}$ uniformly at random. Then*

$$\Pr\left[B[u, \mathcal{U}] = B[v, \mathcal{U}]\right] \geq 4^{-(\ell+1)}.$$

Remark 7.29. *Claim 7.28 assumes that B has no duplicate vertices. This is the reason that we must merge equivalent vertices after each round of restrictions.*

Using Claim 7.28, we now show that a program is indeed approximated by a suffix with $m = \exp(\Theta(\ell))$.

Claim 7.30 ($m \to C^\ell$)**.** *Let $\delta > 0$ and let B be a $(3, i, \ell, m, r)$ program with no unreachable vertices and no duplicate vertices, where $\ell \geq \log\ln(1/\delta)$ and $\ell \geq 1$. Then there exists a value $j \geq i$ such that:*

- *B is a $(3, j, \ell, C^\ell, r)$ program where $C = 32$.*

- $\mathrm{Err}(B, j) \leq \max\{\delta, \mathrm{Err}(B, i)\}$.

Proof. If B is a $(3, i, \ell, C^\ell, r)$ program, then we can take $j = i$. Otherwise, let j be the smallest value such that B is a $(3, j, \ell, C^\ell, r)$ program, and note that each of the C^ℓ ordinary intervals has *at least* one colliding layer. (Otherwise, we could treat it as an unruly layer and decrease j.) By Claim 7.28, when B reads a uniform random input, in each such interval, a collision occurs with probability at least $4^{-(\ell+1)} \geq 4^{-2\ell}$. Therefore,

$$\mathrm{Err}(B, j) \leq (1 - 4^{-2\ell})^{C^\ell} \leq \exp(-(C/16)^\ell) \leq \delta. \qquad \square$$

7.5 Preserving Expectation and Approximation Error: Forbes-Kelley Restrictions

To prove Theorem 7.10, our plan is essentially to alternately apply Claim 7.25 and Claim 7.30 (similar to Meka, Reingold, and Tal [MRT19]) until $\ell = O(\log\log(1/\delta))$ and $m = \mathrm{polylog}(1/\delta)$.[19] At that point, we will be in good shape, because programs with few total colliding layers have low weight. However, two issues remain:

- We need to ensure that our pseudorandom restrictions preserve the expectation of B, as asserted by Theorem 7.10.
- We need to ensure that our pseudorandom restrictions do not significantly increase the approximation error $\mathrm{Err}(B, i)$, so that the layers before layer i do not have too much weight.

As indicated previously, we accomplish both of these goals by using pseudorandom restrictions due to Forbes and Kelley [FK18]. For any constant p, Forbes and Kelley designed a pseudorandom restriction that uses $\widetilde{O}(\log(n/\delta))$ random bits to assign values to approximately a $(1-p)$-fraction of the input variables, while changing the acceptance probability of any constant-width ROBP by at most δ. Furthermore, Forbes and Kelley's pseudorandom restriction is k-wise δ-close to a truly random restriction:

Lemma 7.31 (Forbes-Kelley restrictions [FK18, Lemma 7.2])**.** *Let $p \in (0,1)$ be any constant power of $1/2$. For any $w, n \in \mathbb{N}$ and $\delta > 0$, there is an explicit restriction generator $\mathcal{R} \colon \{0,1\}^s \to \{0,1,\star\}^n$ with seed length $s = \widetilde{O}(w \log(n/\delta))$ such that if we sample $\rho = \mathcal{R}(\mathcal{U}_s)$, then*

1. *If B is a width-w standard-order ROBP, then the restriction ρ preserves the expectation of B up to error δ. That is,*
$$|\mathbb{E}_{\rho,\mathcal{U}}[B|_\rho(\mathcal{U})] - \mathbb{E}[B]| \leq \delta,$$
where $\mathcal{U}$ is sampled uniformly at random and independently of ρ.
2. *The restriction ρ is k-wise (δ/n)-close to $\mathcal{R}_p$ where $k = \log(n/\delta)$.*

Remark 7.32. *Lemma 7.31 as stated above does not literally appear in Forbes and Kelley's work [FK18], but it follows immediately from their analysis. Indeed, they show that if $\mathcal{D}$ is a small-bias distribution and $\mathcal{T}$ is almost k-wise independent, then $\mathcal{D} + \mathcal{T} \wedge \mathcal{U}$ fools B, where $\mathcal{U}$ is uniform random [FK18, Lemma 7.2]. The vectors $\mathcal{D}$ and $\mathcal{T}$ define a restriction where $\mathcal{T}$ indicates the locations of the stars and $\mathcal{D}$ assigns values to the remaining coordinates. The small-bias property implies that this restriction is k-wise close to $\mathcal{R}_{1/2}$ [NN93]. The restriction generator $\mathcal{R}$ samples $O(1)$ copies of this restriction and composes them.*

[19] In contrast, Meka, Reingold, and Tal's analysis [MRT19] roughly corresponds to reaching $\ell = O(\log(1/\delta))$ and $m = \mathrm{poly}(1/\delta)$.

Observe that the the seed length is $\widetilde{O}(\log(n/\delta))$ if w is any constant. Indeed, we will apply the Lemma with $w = 5$, because certain width-5 programs naturally arise in our proof that restrictions do not significantly increase $\mathsf{Err}(B, i)$. To be more precise, we will show that $\mathsf{Err}(B, i)$ does not blow up if we apply a restriction, then apply the $\mathsf{MergeEquiv}(\cdot)$ and $\mathsf{DelUnreach}(\cdot)$ operators:

Claim 7.33. *Let B be a width-3 length-n standard-order ROBP. Let $\rho = \mathcal{R}(\mathcal{U}_s) \in \{0, 1, \star\}^n$ be the pseudorandom restriction from Lemma 7.31 with $w = 5$ and any values for p and δ. Let $B' = \mathsf{DelUnreach}(\mathsf{MergeEquiv}(B|_\rho))$. Then for any $i \in \{0, \dots, n\}$,*

$$\mathbb{E}[\mathrm{Err}(B', i)] \leq \mathrm{Err}(B, i) + \delta.$$

Proof. Merging equivalent vertices has no effect on Err, and deleting unreachable vertices can only decrease Err, so $\mathrm{Err}(B', i) \leq \mathrm{Err}(B|_\rho, i)$. Let g be the predicate

$$g(x) = 1 \iff \text{there exist } u, v \in V^{(i)} \text{ such that } B^{\leftarrow u}(x) \neq B^{\leftarrow v}(x).$$

The function g can be computed by a standard-order ROBP that simulates B from all possible start vertices in $V^{(i)}$ simultaneously. At any given moment, each of the $|V^{(i)}|$ computations is in one of up to three possible states, so we can compute g by a program of width $3^3 = 27$. We can be more economical by observing that we only need to keep track of the *unordered set* of current states, and furthermore, all singleton sets can be handled by a single "reject" state. Thus, the width is at most $\binom{3}{3} + \binom{3}{2} + 1 = 5$. Consequently, ρ preserves the expectation of g up to error δ. Therefore,

$$\mathbb{E}_\rho[\mathrm{Err}(B|_\rho, i)] = \mathbb{E}_{\rho,\mathcal{U}}[g|_\rho(\mathcal{U})] \leq \mathbb{E}[g] + \delta = \mathrm{Err}(B, i) + \delta. \qquad \square$$

7.6 Composing Several Rounds of Restrictions

To prove Theorem 7.10, we will iteratively alternate between applying a Forbes-Kelley restriction and applying the $\mathsf{MergeEquiv}(\cdot)$ and $\mathsf{DelUnreach}(\cdot)$ operators. The following lemma will help us to reason about the overall impact of these $\mathsf{MergeEquiv}(\cdot)$ and $\mathsf{DelUnreach}(\cdot)$ operators. The lemma says that the program $\mathsf{MergeEquiv}(\mathsf{DelUnreach}(B))$ only depends on the functional behavior of B, not its graphical structure. Indeed, in some sense, $\mathsf{MergeEquiv}(\mathsf{DelUnreach}(B))$ is the unique "minimal" ROBP computing whatever function B computes.

Lemma 7.34 (Characterizing $\mathsf{MergeEquiv}(\mathsf{DelUnreach}(B))$)**.** *Let B be a standard-order ROBP, and let $B' = \mathsf{MergeEquiv}(\mathsf{DelUnreach}(B))$. For each $0 \leq i \leq n$, define an equivalence relation $\sim$ on $\{0,1\}^i$ by the rule*

$$x \sim y \iff \forall z,\ B(xz) = B(yz).$$

Let $[x]$ denote the equivalence class containing the string x. Define another standard-order ROBP B'' as follows. Layer i consists of the equivalence classes into which $\{0,1\}^i$ is partitioned by $\sim$. The outgoing edge from vertex $[x]$ with label b leads to the vertex $[xb]$. In the final layer, $[x]$ is an accepting state if and only if $B(x) = 1$. Then B' is isomorphic to B'', i.e., one can be obtained from the other by renaming vertices.

Proof. For each vertex $[x]$ of B'', let $\phi([x])$ be the vertex of B' that is reached by reading the string x. This is well-defined, because if $x \sim y$, then $\mathsf{MergeEquiv}(\cdot)$ merges the two vertices reached by x and by y. The function ϕ is injective, because B' computes the same function as B. The function ϕ is surjective, because every vertex v in B' is reachable, hence there is some string x such that $\phi([x]) = v$. Finally, it is clear that ϕ preserves the outgoing edges of the programs, the start vertex, and the set of accepting vertices. $\square$

Claim 7.35 ($\ell \to \log\log(1/\delta)$)**.** *Let B be a width-3 length-n ROBP. Let $\rho^0, \dots, \rho^t$ be independent copies of the pseudorandom restriction from Lemma 7.31 with $w = 5$, with p a small enough constant, and with $\delta \le 1/(\log\log n)^5$, for a certain value $t = O(\log\log(n/\delta))$. Let $\rho = \rho^t \circ \cdots \circ \rho^0$ and $B' = \mathsf{MergeEquiv}(\mathsf{DelUnreach}(B|_\rho))$. Then except with probability $\delta^{1/4}$, there is a value i such that:*

- *B' is a $(3, i, \ell, m, r)$ program where $\ell = O(\log\log(1/\delta))$, $m = O(\log^5(1/\delta))$, and $r = O(\log(1/\delta) \cdot \log\log(n/\delta))$.*
- *$\mathrm{Err}(B', i) \le \delta^{1/4}$.*

Proof. Let $C = 32$, let $\widehat{B} = \mathsf{DelUnreach}(\mathsf{MergeEquiv}(B))$, and let $B^0 = \mathsf{MergeEquiv}(\mathsf{DelUnreach}(\widehat{B}|_{\rho^0}))$. In the program $\widehat{B}$, the first layer has width 1 and the last layer has width at most 2. Therefore, by Claim 7.16 and Claim 7.20, except with probability 2δ, B^0 is a $(3, i_0, \ell_0, C^{\ell_0}, r_0)$ program with $i_0 = 0$, $\ell_0 = O(\log(n/\delta))$, and $r_0 = 0$. Trivially, $\mathrm{Err}(B^0, i_0) = 0$.

In each subsequent round, we start with a program B^{j-1} that is a $(3, i_{j-1}, \ell_{j-1}, C^{\ell_{j-1}}, r_{j-1})$ program and that has no duplicate vertices. Let $B^j = \mathsf{DelUnreach}(\mathsf{MergeEquiv}(B^{j-1}|_{\rho^j}))$. Let $\ell_j = \ell_{j-1}/2$ and $r_j = r_{j-1} + \log(1/\delta)$. By Claim 7.25, except with probability $\sqrt{\delta/n}$, the program $\mathsf{MergeEquiv}(B^{j-1}|_{\rho^j})$ is a $(3, i_{j-1}, \ell_j, m', r_j)$ program for some m'. Deleting unreachable vertices cannot create new colliding layers, so B^j is also a $(3, i_{j-1}, \ell_j, m', r_j)$ program. This program B^j has no duplicate vertices and no unreachable vertices, so we may apply Claim 7.30 to conclude that there exists some i_j such that B^j is a $(3, i_j, \ell_j, C^{\ell_j}, r_j)$ program, and furthermore

$$\mathrm{Err}(B^j, i_j)) \le \max\{\delta, \mathrm{Err}(B^j, i_{j-1}))\} \le \mathrm{Err}(B^j, i_{j-1}) + \delta.$$

(In the unlikely event that $\mathsf{MergeEquiv}(B^{j-1}|_{\rho^j}$ is *not* a $(3, i_{j-1}, \ell_j, m', r_j)$ program, then we define $i_j = i_{j-1}$. This ensures that the equation above still holds.) We terminate the process after t rounds, where t is the largest value such that $\ell_t \ge \log\ln(1/\delta)$. (That way, the hypotheses of Claim 7.30 are always satisfied.)

Overall, except with probability $2\delta + t\sqrt{\delta/n}$, the program B^t is a $(3, i, \ell, m, r)$ program, where $i = i_t$, $\ell \le 2\log\ln(1/\delta)$, $m \le C^\ell \le (\ln(1/\delta))^5$, and $r = r_t \le t\log(1/\delta)$. Furthermore,

$$\mathbb{E}[\mathrm{Err}(B^t, i_t)] \le \delta + \mathbb{E}[\mathrm{Err}(B^t, i_{t-1})] \le 2\delta + \mathbb{E}[\mathrm{Err}(B^{t-1}, i_{t-1})] \le \cdots \le 2t\delta.$$

Therefore, by Markov's inequality, except with probability $\sqrt{2t\delta}$, we have $\mathrm{Err}(B^t, i_t) \le \sqrt{2t\delta}$. Since $2\delta + t\sqrt{\delta/n} + \sqrt{2t\delta} < \delta^{1/4}$, the program B^t satisfies the conclusions of the claim.

Finally, we claim that the programs B^t and B' are isomorphic. To see why, observe first that $B|_\rho$ and $B^{t-1}|_{\rho^t}$ compute the same function $f\colon \{0,1\}^n \to \{0,1\}$, because neither the $\mathsf{DelUnreach}(\cdot)$ operator nor the $\mathsf{MergeEquiv}(\cdot)$ operator affects the function that is computed by the program. Next, observe that even though we defined B^t as $\mathsf{DelUnreach}(\mathsf{MergeEquiv}(B^{t-1}|_{\rho^t})$, we can equally well say that $B^t = \mathsf{MergeEquiv}(\mathsf{DelUnreach}(B^{t-1}|_{\rho^t}))$, because applying MergeEquiv and then DelUnreach is equivalent to applying DelUnreach followed by MergeEquiv. To conclude, observe that by Lemma 7.34, B' and B^t are both isomorphic to a program that is defined only based on the input-output behavior of f. □

7.7 Few Colliding Layers Implies Bounded Weight

We are nearing the end of the proof of Theorem 7.10. The conclusion of Claim 7.35 says that B' is a $(3, i, \ell, m, r)$ program where ℓ, m, r, and $\mathrm{Err}(B', i)$ are small. We now show that it follows that B' has low weight.

Claim 7.36 (Few colliding layers $\implies$ low weight)**.** *Let B be a (w, i, ℓ, m, r) program. Then*

$$\mathsf{Weight}(B) \leq 2w \cdot i \cdot \mathrm{Err}(B, i) + O(w^2 \cdot (\ell m + r + 1)).$$

Proof. Let the layers of B be $V^{(0)}, \ldots, V^{(n)}$. Let $e = (u, v)$ be an edge where $v \in V^{(1)} \cup \cdots \cup V^{(i)}$. Then

$$\mathsf{Weight}(e) = |\mathbb{E}[B^{\leftarrow u}] - \mathbb{E}[B^{\leftarrow v}| \leq \Pr_{\mathcal{U}}[B^{\leftarrow u}(\mathcal{U}) \neq B^{\leftarrow v}(\mathcal{U})] \leq \mathrm{Err}(B, i),$$

where $\mathcal{U} \in \{0,1\}^n$ is selected uniformly at random. Therefore, the sum of weights of edges prior to $V^{(i)}$ is at most $2w \cdot i \cdot \mathrm{Err}(B, i)$.

There are at most $\ell m + r$ colliding layers in the interval $(i, n]$. Each of these colliding layers trivially has weight at most $2w$. Finally, consider a region between two colliding layers. Say layers $j, j+1, \ldots, k$ are all non-colliding. Then the results of Braverman, Rao, Raz, and Yehudayoff [BRRY14] show that the total weight of these layers is at most $O(w^2)$.[20] The claim follows. □

Combining Claim 7.35 with Claim 7.36 shows that with high probability, the program $\mathsf{MergeEquiv}(\mathsf{DelUnreach}(B|_\rho))$ has low weight. The following claim helps us conclude that $\mathsf{DelUnreach}(B|_\rho)$ has low weight (even without merging vertices).

Claim 7.37. *Let B be a width-w standard-order ROBP. Then*

$$\mathsf{Weight}(B) \leq w \cdot \mathsf{Weight}(\mathsf{MergeEquiv}(B)).$$

Proof. Let $b \in \{0,1\}$, let $i \in [n]$, and let (u, v) be an edge from layer $i-1$ to layer i of B. The $\mathsf{MergeEquiv}(\cdot)$ operator merges u with all vertices in layer $i-1$ that are equivalent to u, leading to a vertex u'. Let (u', v') be the outgoing edge of u' in $\mathsf{MergeEquiv}(B)$ that is labeled b. Then

$$\mathbb{E}[B^{\leftarrow u}] = \mathbb{E}[\mathsf{MergeEquiv}(B)^{\leftarrow u'}]$$

and

$$\mathbb{E}[B^{\leftarrow v}] = \mathbb{E}[\mathsf{MergeEquiv}(B)^{\leftarrow v'}],$$

so $\mathsf{Weight}((u, v)) = \mathsf{Weight}((u', v'))$. Therefore, if we let S be the set of edges leading to layer i of B with label b, and we let T be the set of edges leading to layer i of $\mathsf{MergeEquiv}(B)$ with label b, then there is a function $f \colon S \to T$ such that

$$\sum_{e \in S} \mathsf{Weight}(e) = \sum_{e \in S} \mathsf{Weight}(f(e)) \leq |S| \cdot \max_{e \in T} \mathsf{Weight}(e) \leq w \cdot \sum_{e \in T} \mathsf{Weight}(e).$$

Summing over all i and b completes the proof. □

Putting everything together proves Theorem 7.10, as we now explain.

Proof of Theorem 7.10. The restriction generator $\mathcal{R}$ samples t independent copies $\rho^0, \ldots, \rho^t$ of the pseudorandom restriction from Lemma 7.31 with $w = 5$, with p a small enough constant, and with $\delta = (\varepsilon/n)^4$, where $t = O(\log\log(n/\varepsilon))$ is the value from Claim 7.35. Then, $\mathcal{R}$ outputs their composition: $\rho = \rho^t \circ \cdots \circ \rho^0$.

[20] Label each vertex v with the acceptance probability from that vertex, $\mathbb{E}[B^{\leftarrow v}]$. In this way, we can view layers $j, j+1, \ldots, k$ as a regular "evaluation program" of length $k - j$, to use the language of Braverman, Rao, Raz, and Yehudayoff [BRRY14]. All of the vertex labels are in the interval $[0, 1]$, so this regular evaluation program has weight $O(w^2)$ [BRRY14, Lemma 6].

By Lemma 7.31, each restriction ρ^i preserves the expectation of B up to error δ. Therefore, ρ preserves the expectation of B up to error $t\delta < \varepsilon$. To show the second conclusion of Theorem 7.10, let $B' = \mathsf{MergeEquiv}(\mathsf{DelUnreach}(B|_\rho))$. By Claim 7.35, with probability $1-\varepsilon$, there is a value i such that B' is a (w, i, ℓ, m, r) program where $w = 3$, $\ell = O(\log\log(n/\varepsilon))$, $m = O(\log^5(n/\varepsilon))$, and $r = \widetilde{O}(\log(n/\varepsilon))$, and furthermore, $\mathrm{Err}(B', i) \le \varepsilon/n$. Consequently, by Claim 7.36, we have

$$\mathsf{Weight}(B') \le 2w \cdot i \cdot \mathrm{Err}(B', i) + O(w^2 \cdot (\ell m + r + 1)) = \widetilde{O}(\log^5(n/\varepsilon)).$$

Finally, by Claim 7.37,

$$\mathsf{Weight}(\mathsf{DelUnreach}(B|_\rho)) \le w \cdot \mathsf{Weight}(B') = \widetilde{O}(\log^5(n/\varepsilon)). \qquad \square$$

Remark 7.38. *Theorem 7.10 guarantees that with high probability over ρ, we have* $\mathsf{Weight}(\mathsf{DelUnreach}(B|_\rho)) \le \mathrm{polylog}(n/\varepsilon)$. *The proof actually shows something stronger, which is that with high probability over ρ, every standard-order ROBP B' that computes the function $B|_\rho$ satisfies* $\mathsf{Weight}(\mathsf{DelUnreach}(B')) \le w \cdot \mathrm{polylog}(n/\varepsilon)$, *where w is the width of B'.*

8 Simplified Non-Black-Box Derandomization of Regular ROBPs

In this section, we give a relatively elementary proof of the main result in Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's work [AKMPSV20] (Theorem 1.10). As a reminder, they gave a nearly-logarithmic space algorithm for estimating the acceptance probability of a given regular standard-order ROBP.

As explained in Section 3, our proof is based on the concept of *singular-value approximation*, introduced by Ahmadinejad, Peebles, Pyne, Sidford, and Vadhan [APPSV23].

Reminder of Definition 3.1. *Let $\widetilde{\mathbf{W}}, \mathbf{W} \in \mathbb{R}^{w\times w}$ be doubly stochastic matrices. We say $\widetilde{\mathbf{W}}$ τ-singular-value approximates $\mathbf{W}$, denoted as $\widetilde{\mathbf{W}} \overset{\mathrm{sv}}{\approx}_\tau \mathbf{W}$, if for every $x, y \in \mathbb{R}^w$,*

$$\left| y^T \left(\widetilde{\mathbf{W}} - \mathbf{W}\right) x \right| \le \frac{\tau}{4} \cdot \left(\|x\|^2_{\mathbf{I}-\mathbf{W}^T\mathbf{W}} + \|y\|^2_{\mathbf{I}-\mathbf{W}\mathbf{W}^T} \right).$$

Our new proof of Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's main result [AKMPSV20] can be decomposed into the following two steps. The first step is to show that a recursive application of (a version of) Rozenman and Vadhan's "derandomized squaring" operation [RV05] computes a singular-value approximation to the random walk matrix of the given regular ROBP in $\widetilde{O}(\log n)$ space with $1/\mathrm{polylog}(n)$ error. Formally:

Theorem 8.1 (Space-efficient moderate-error sv-approximation)**.** *There is a deterministic algorithm that, given a regular width-w length-n standard-order ROBP B, outputs a matrix $\widetilde{\mathbf{W}}_{n\leftarrow 0} \in \mathbb{R}^{w\times w}$ such that $\widetilde{\mathbf{W}}_{n\leftarrow 0} \overset{\mathrm{sv}}{\approx}_{\tau_0} \mathbf{W}_{n\leftarrow 0}$, where $\mathbf{W}_{n\leftarrow 0}$ is defined from B as in Section 4 and $\tau_0 = 1/(64 \cdot \log^2 n)$. Furthermore, the algorithm uses $\widetilde{O}(\log(nw))$ bits of space.*

Theorem 8.1 readily follows from the analysis by Ahmadinejad, Peebles, Pyne, Sidford, and Vadhan [APPSV23]. For the sake of completeness, we include our own proof of Theorem 8.1, but we defer it to Appendix A. By applying Theorem 8.1 once for each edge $(\ell, r) \in E(\mathsf{SC}_n)$, we can construct[21] an approximation ensemble $\widetilde{\mathcal{W}}$ that has moderate error in the following "singular value" sense:

[21] That is, to construct $\widetilde{\mathbf{W}}_{r\leftarrow\ell}$, we apply Theorem 8.1 to the branching program obtained from B by replacing the edges prior to layer ℓ and subsequent to layer r with trivial identity edges.

Definition 8.2 (τ-sv-accurate)**.** *Let B be a width-w length-n standard-order regular ROBP, and let $\widetilde{\mathcal{W}} = \{\widetilde{\mathbf{W}}_{r\leftarrow\ell} : (\ell, r) \in E(\mathsf{SC}_n)\}$ be an approximation ensemble. We say $\widetilde{\mathcal{W}}$ is* τ-sv-accurate *(with respect to B), if for every $(\ell, r) \in E(\mathsf{SC}_n)$, the matrix $\widetilde{\mathbf{W}}_{r\leftarrow\ell}$ is doubly stochastic, and $\widetilde{\mathbf{W}}_{r\leftarrow\ell} \overset{\text{sv}}{\approx}_\tau \mathbf{W}_{r\leftarrow\ell}$, where $\mathbf{W}_{r\leftarrow\ell}$ is defined from B as in Section 4.*

The second step of our proof of Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's main result [AKMPSV20] is an error reduction procedure. Given a regular ROBP B, and given an approximation ensemble that is τ-sv-accurate with a moderate error $\tau = 1/\text{polylog}(n)$, we show how to compute the acceptance probability $\mathbb{E}[B]$ to within any tiny error $\pm\varepsilon$, using space that is *doubly logarithmic* in $1/\varepsilon$.

Theorem 8.3 (Non-black-box error reduction for regular ROBPs)**.** *Let B be a width-w length-n standard-order regular ROBP. Let $\mathcal{W}$ be an approximation ensemble that is τ_0-sv-accurate, where $\tau_0 = 1/(64 \cdot \log^2 n)$. There is a deterministic algorithm that, given B, $\widetilde{\mathcal{W}}$, and $\varepsilon \in (0, 1)$, outputs a number that is within $\pm\varepsilon$ of $\mathbb{E}[B]$. The algorithm uses $O(\log(nw) \cdot \log\log(nw/\varepsilon))$ bits of space.*

Combining the two algorithms above gives the desired algorithm for estimating the acceptance probability of a given regular ROBP, i.e., it proves Theorem 1.10, which we restate below as a reminder.

Theorem 8.4 (Theorem 1.10, restated)**.** *There is a deterministic algorithm that uses $\widetilde{O}(\log(wn) \cdot \log\log(1/\varepsilon))$ bits of space and outputs a value that is within $\pm\varepsilon$ of $\mathbb{E}[B]$, given a width-w length-n standard-order regular ROBP B and a value $\varepsilon \in (0, 1)$ as inputs.*

The remainder of this section consists of the proof of Theorem 8.3.

8.1 The F-Seminorm

The proof of Theorem 8.3 uses our error reduction framework described in Section 4. As explained in Section 4, the regular branching program B has a Laplacian matrix $\mathbf{L}$, and the ensemble $\widetilde{\mathcal{W}}$ induces matrices $\mathbf{A}_m$, $\Delta\mathbf{W}$, etc. For convenience, we will assume that the length-one walks in $\widetilde{\mathcal{W}}$ have zero error, i.e., $\widetilde{\mathbf{W}}_{j+1\leftarrow j} = \mathbf{W}_{j+1\leftarrow j}$ for each $j \in \{0, \ldots, n-1\}$. This assumption is without loss of generality, because we can compute $\mathbf{W}_{j+1\leftarrow j}$ from B exactly using $O(\log(nw))$ bits of space.

In order to use our error reduction framework, we must show that $\|\mathbf{L}^{-1}\Delta\mathbf{W}\|$ is moderately small, where $\|\cdot\|$ is some extended submultiplicative matrix seminorm (see Theorem 4.4). In Section 6, we chose $\|\cdot\| = \|\cdot\|_\infty$. In this section, we will use a more sophisticated seminorm, the "**F**-seminorm," which takes into account the structure of the branching program B.

Toward defining the **F**-seminorm, let $\Phi\colon \mathbb{R}^w \to \mathbb{R}_{\geq 0}$ be the function

$$\Phi(u) := \|u\|_2^2.$$

We use this notation $\Phi(u)$ because we think of $\Phi(u)$ as a "potential function." This perspective will be crucial for our analysis. For a vector $u \in \mathbb{R}^w$ and $0 \leq \ell \leq r \leq n$, we define

$$\|u\|_{\ell\searrow r}^2 = \|u\|_{\mathbf{I}-\mathbf{W}_{r\leftarrow\ell}^T\mathbf{W}_{r\leftarrow\ell}}^2 = \Phi(u) - \Phi(\mathbf{W}_{r\leftarrow\ell} \cdot u).$$

Intuitively, the quantity above measures the drop of potential starting from the ℓ-th layer with vector u and moving along $\mathbf{W}_{r\leftarrow\ell}$ to arrive at the r-th layer with vector $\mathbf{W}_{r\leftarrow\ell} \cdot u$. Note that because B is a *regular* branching program, the matrix $\mathbf{W}_{r\leftarrow\ell}$ is doubly stochastic, and hence $\|\cdot\|_{\ell\searrow r}$ is indeed a seminorm.

Let $x \in \mathbb{R}^{(n+1)w}$, and write it as a block vector $x = (x^{[0]}, \ldots, x^{[n]})$, where $x^{[j]} \in \mathbb{R}^{w \times w}$ is the restriction of x to the j-th layer of the branching program B. Recall from Definition 4.1 that U_k is the set of multiples of 2^k up to n, i.e.,

$$U_k = \{i \cdot 2^k : i \in \mathbb{Z} \cap [0, n/2^k]\} = \{0, 2^k, 2 \cdot 2^k, 3 \cdot 2^k, \ldots, n\}.$$

We define the $\mathbf{F}_k$-seminorm of x as

$$\|x\|_{\mathbf{F}_k}^2 = \sum_{j \in U_k} \left\| x^{[j]} \right\|_{j \searrow j+2^k}^2.$$

In particular, in the corner case when $j = n$, we let $\mathbf{W}_{n+2^k \leftarrow n} = \mathbf{0}$, so the term is simply $\|x^{[n]}\|_2^2$. By the equation above, we see that $\|x\|_{\mathbf{F}_k}^2$ measures the total amount of potential drop made in 2^k steps from all the sub-vectors of x on the layers in U_k.

Next, we define the $\mathbf{F}$-seminorm of a vector x as[22]

$$\|x\|_{\mathbf{F}}^2 = \sum_{k=1}^{\log n} \|x\|_{\mathbf{F}_k}^2.$$

Finally, this vector seminorm $\|\cdot\|_{\mathbf{F}}$ induces an extended submultiplicative matrix seminorm $\|\cdot\|_{\mathbf{F}}$, as explained in Definition 2.4:

$$\|\mathbf{M}\|_{\mathbf{F}} = \min\{\lambda \in \mathbb{R} \cup \{\infty\} : \forall x, \|\mathbf{M}x\|_{\mathbf{F}} \le \lambda \cdot \|x\|_{\mathbf{F}}\}.$$

Our main job is to prove the following lemma.

Lemma 8.5 (Main Lemma)**.** *Let B be a width-w length-n standard-order regular ROBP. Let $\widetilde{\mathcal{W}}$ be a τ-sv-accurate approximation ensemble, where $\tau \in (0, \frac{1}{64 \cdot \log^2 n}]$. Let $\mathbf{L}$ and $\Delta\mathbf{W}$ be defined from B and $\widetilde{\mathcal{W}}$ as in Section 4. Then*

$$\left\| \mathbf{L}^{-1} \Delta \mathbf{W} \right\|_{\mathbf{F}}^2 \le 4 \cdot \log^2 n \cdot \tau.$$

Lemma 8.5 enables us to use our error reduction framework. After proving Lemma 8.5, we will show that this error reduction framework implies Theorem 8.3.

8.2 Bounding $\|\mathbf{L}^{-1}\Delta\mathbf{W}^{(i)}x\|_{\mathbf{F}_k}$ When $i \le k$

The proof of Lemma 8.5 relies on the following simple lemma regarding singular-value approximation, whose proof can be found in Appendix C.

Lemma 8.6. *Let $\mathbf{W}_1, \mathbf{W}_2, \widetilde{\mathbf{W}}_1, \widetilde{\mathbf{W}}_2, \widetilde{\mathbf{W}}$ be doubly stochastic matrices. Let $\mathbf{W} = \mathbf{W}_2\mathbf{W}_1$. Assume that $\widetilde{\mathbf{W}}_1 \overset{\text{sv}}{\approx}_\varepsilon \mathbf{W}_1$, $\widetilde{\mathbf{W}}_2 \overset{\text{sv}}{\approx}_\varepsilon \mathbf{W}_2$, and $\widetilde{\mathbf{W}} \overset{\text{sv}}{\approx}_\varepsilon \mathbf{W}$. Then*

$$\mathbf{W}_2\mathbf{W}_1 + (\widetilde{\mathbf{W}} - \widetilde{\mathbf{W}}_2\widetilde{\mathbf{W}}_1) \overset{\text{sv}}{\approx}_{2\varepsilon(1+\varepsilon/4)} \mathbf{W}_2\mathbf{W}_1.$$

In other words, for all vectors x, y,

$$\left| y^T \left(\widetilde{\mathbf{W}} - \widetilde{\mathbf{W}}_2\widetilde{\mathbf{W}}_1 \right) x \right| \le \frac{\varepsilon \cdot (1 + \varepsilon/4)}{2} \cdot \left(\|x\|_{\mathbf{I} - \mathbf{W}^T\mathbf{W}}^2 + \|y\|_{\mathbf{I} - \mathbf{W}\mathbf{W}^T}^2 \right).$$

[22] In the definition of the $\mathbf{F}$-seminorm, we do not include a $k = 0$ term. This is essentially for convenience, and it corresponds to our assumption that $\widetilde{\mathbf{W}}_{j+1 \leftarrow j} = \mathbf{W}_{j+1 \leftarrow j}$ for each j.

The proof of Lemma 8.5 also relies on the following simple fact, which says that when taking two consecutive steps of a random walk using transition matrices $\mathbf{W}$ and $\mathbf{W}^T$, the decrease in potential of the latter step is always smaller than that of the former.

Fact 8.7. *For any matrix $\mathbf{W} \in \mathbb{R}^{w\times w}$ and $z \in \mathbb{R}^w$, we have*

$$\Phi(z) - \Phi(\mathbf{W}z) \geq \Phi(\mathbf{W}z) - \Phi(\mathbf{W}^T\mathbf{W}z).$$

Note that if $\mathbf{W}$ is doubly stochastic, the inequality above can also be written as

$$\|z\|^2_{\mathbf{I}-\mathbf{W}^T\mathbf{W}} \geq \|\mathbf{W}z\|^2_{\mathbf{I}-\mathbf{W}\mathbf{W}^T}.$$

Proof. Because $\mathbf{I} - \mathbf{W}^T\mathbf{W}$ is symmetric, we have[23]

$$\begin{aligned}(\Phi(z) - \Phi(\mathbf{W}z)) - (\Phi(\mathbf{W}z) - \Phi(\mathbf{W}^T\mathbf{W}z)) &= z^T\Big[(\mathbf{I} - \mathbf{W}^T\mathbf{W}) - (\mathbf{W}^T\mathbf{W} - (\mathbf{W}^T\mathbf{W})^2)\Big]z \\ &= z^T(\mathbf{I} - \mathbf{W}^T\mathbf{W})^2 z \\ &= \Big\|(\mathbf{I} - \mathbf{W}^T\mathbf{W}) \cdot z\Big\|_2^2 \geq 0.\end{aligned}$$

□

The core of the proof of Lemma 8.5 is the following variant of Lemma 8.5, in which we focus on the "level-i" and "level-k" edges in the shortcut graph SC_n.

Lemma 8.8. *Let B and $\widetilde{\mathcal{W}}$ be as in Lemma 8.5. Let $\mathbf{L}^{-1}$ and $\Delta\mathbf{W}^{(i)}$ be defined as in Section 4. For every $i, k \in [\log n]$ and every $x \in \mathbb{R}^{(n+1)w}$, we have*

$$\Big\|\mathbf{L}^{-1}\Delta\mathbf{W}^{(i)}x\Big\|^2_{\mathbf{F}_k} \leq 4\tau \cdot \|x\|^2_{\mathbf{F}_i}.$$

In the remainder of this subsection, we focus on proving Lemma 8.8 in the special case $i \leq k$. We handle the case $i > k$ in the next subsection.

Proof of Lemma 8.8 assuming $i \leq k$. Let $z = \mathbf{L}^{-1}\Delta W^{(i)}x$, so our goal is to bound $\|z\|^2_{\mathbf{F}_k}$. Like before, we write $x = (x^{[0]}, \ldots, x^{[n]})$, where $x^{[i]}$ is the restriction of x to layer i of B, and similarly $z = (z^{[0]}, \ldots, z^{[n]})$. We observe the following recursion formula: for every $j \in U_k \setminus \{n\}$, we have

$$z^{[j+2^k]} = \mathbf{W}_{j+2^k\leftarrow j} \cdot z^{[j]} + \sum_{\ell\in U_i\cap[j,j+2^k)} \mathbf{W}_{j+2^k\leftarrow\ell+2^i} \cdot \Delta\mathbf{W}^{(i)}_{(\ell+2^i)\leftarrow\ell} x^{[\ell]}.$$

For convenience, we define $\mathbf{W}_{n+2^k\leftarrow n}$ to be the zero matrix and $z^{[n+2^k]}$ to be the zero vector, so the equation above holds even for $j = n$. For each layer $j \in U_k$, we define

$$\widetilde{z}^{[j+2^k]} = \mathbf{W}_{j+2^k\leftarrow j} \cdot z^{[j]} \qquad \text{and} \qquad s^{[j+2^k]} = \sum_{\ell\in U_i\cap[j,j+2^k)} \mathbf{W}_{j+2^k\leftarrow\ell+2^i} \cdot \Delta\mathbf{W}^{(i)}_{(\ell+2^i)\leftarrow\ell} x^{[\ell]},$$

so that $z^{[j+2^k]} = \widetilde{z}^{[j+2^k]} + s^{[j+2^k]}$. Then by definition,

$$\begin{aligned}\|z\|^2_{\mathbf{F}_k} = \sum_{j\in U_k} \|z^{[j]}\|^2_{j\searrow j+2^k} &= \sum_{j\in U_k} \underbrace{\Big(\Phi(z^{[j]}) - \Phi(\widetilde{z}^{[j+2^k]})\Big)}_{(*)} \\ &= \Phi(z^{[0]}) - \Phi(z^{[n+2^k]}) + \sum_{j\in U_k}\Big(\Phi(z^{[j+2^k]}) - \Phi(\widetilde{z}^{[j+2^k]})\Big) \\ &= \sum_{j\in U_k} \underbrace{\Big(\Phi(z^{[j+2^k]}) - \Phi(\widetilde{z}^{[j+2^k]})\Big)}_{(**)}. \qquad (15)\end{aligned}$$

[23] We used ChatGPT as part of our process for developing this proof.

(Quantity $(*)$ is the decrease in potential at step j. We think of quantity $(**)$ as the amount of error in step j. See Figure 3.) Fix $j \in U_k$. We now turn to bounding quantity $(**)$:

$$\Phi(z^{[j+2^k]}) = \Phi(\widetilde{z}^{[j+2^k]} + s^{[j+2^k]}) = \Phi(\widetilde{z}^{[j+2^k]}) + 2\langle \widetilde{z}^{[j+2^k]}, s^{[j+2^k]} \rangle + \langle s^{[j+2^k]}, s^{[j+2^k]} \rangle. \tag{16}$$

To bound the last two terms, we use the following claim:

Claim 8.9. *For every $u \in \mathbb{R}^w$, we have*

$$\left| \langle u, s^{[j+2^k]} \rangle \right| \leq \tau \cdot \|u\|^2_{\mathbf{I} - \mathbf{W}_{j+2^k \leftarrow j} \mathbf{W}^T_{j+2^k \leftarrow j}} + \tau \cdot \sum_{\ell \in U_i \cap [j, j+2^k)} \left\| x^{[\ell]} \right\|^2_{\ell \searrow \ell + 2^i}.$$

Proof. By the definition of $s^{[j+2^k]}$,

$$\begin{aligned} \left| \langle u, s^{[j+2^k]} \rangle \right| &= \left| \sum_{\ell \in U_i \cap [j, j+2^k)} (u^T \mathbf{W}_{j+2^k \leftarrow \ell+2^i}) \cdot (\Delta \mathbf{W}^{(i)})_{(\ell+2^i) \leftarrow \ell} \cdot x^{[\ell]} \right| \\ &\leq \sum_{\ell \in U_i \cap [j, j+2^k)} \left| (u^T \mathbf{W}_{j+2^k \leftarrow \ell+2^i}) \cdot (\Delta \mathbf{W}^{(i)})_{(\ell+2^i) \leftarrow \ell} \cdot x^{[\ell]} \right| \end{aligned}$$

Since $\mathcal{W}$ is τ-sv-accurate, and $\tau \leq \frac{1}{64}$, by Lemma 8.6,

$$\left| \langle u, s^{[j+2^k]} \rangle \right| \leq \tau \cdot \left(\sum_{\ell \in U_i \cap [j, j+2^k)} \|\mathbf{W}^T_{j+2^k \leftarrow \ell+2^i} u\|^2_{\mathbf{I} - \mathbf{W}_{\ell+2^i \leftarrow \ell} \mathbf{W}^T_{\ell+2^i \leftarrow \ell}} + \sum_{\ell \in U_i \cap [j, j+2^k)} \|x^{[\ell]}\|^2_{\ell \searrow \ell + 2^i} \right)$$

We further observe that the former sum in the right-hand side is telescopic:

$$\sum_{\ell \in U_i \cap [j, j+2^k)} \|\mathbf{W}^T_{j+2^k \leftarrow \ell+2^i} u\|^2_{\mathbf{I} - \mathbf{W}_{\ell+2^i \leftarrow \ell} \mathbf{W}^T_{\ell+2^i \leftarrow \ell}} = \|u\|^2_{\mathbf{I} - \mathbf{W}_{j+2^k \leftarrow j} \mathbf{W}^T_{j+2^k \leftarrow j}},$$

which completes the proof of Claim 8.9. □

We now continue the proof of Lemma 8.8 where we left off (Equation 16). By Claim 8.9 with $u = \widetilde{z}^{[j+2^k]}$, we have

$$\begin{aligned} 2|\langle \widetilde{z}^{[j+2^k]}, s^{[j+2^k]} \rangle| &\leq 2\tau \cdot \|\widetilde{z}^{[j+2^k]}\|^2_{\mathbf{I} - \mathbf{W}_{j+2^k \leftarrow j} \mathbf{W}^T_{j+2^k \leftarrow j}} + 2\tau \cdot \sum_{\ell \in U_i \cap [j, j+2^k)} \left\| x^{[\ell]} \right\|^2_{\ell \searrow \ell + 2^i} \\ &= 2\tau \cdot \|\mathbf{W}_{j+2^k \leftarrow j} \cdot z^{[j]}\|^2_{\mathbf{I} - \mathbf{W}_{j+2^k \leftarrow j} \mathbf{W}^T_{j+2^k \leftarrow j}} + 2\tau \cdot \sum_{\ell \in U_i \cap [j, j+2^k)} \left\| x^{[\ell]} \right\|^2_{\ell \searrow \ell + 2^i} \\ &\qquad \text{(Def. of } \widetilde{z}^{[j+2^k]}\text{)} \\ &\leq 2\tau \cdot \|z^{[j]}\|^2_{j \searrow j + 2^k} + 2\tau \cdot \sum_{\ell \in U_i \cap [j, j+2^k)} \left\| x^{[\ell]} \right\|^2_{\ell \searrow \ell + 2^i}. \qquad \text{(Fact 8.7)} \end{aligned}$$

We also use Claim 8.9 on $u = s^{[j+2^k]}$ to obtain

$$\begin{aligned} \langle s^{[j+2^k]}, s^{[j+2^k]} \rangle &\leq \tau \cdot \|s^{[j+2^k]}\|^2_{\mathbf{I} - \mathbf{W}_{j+2^k \leftarrow j} \mathbf{W}^T_{j+2^k \leftarrow j}} + \tau \cdot \sum_{\ell \in U_i \cap [j, j+2^k)} \left\| x^{[\ell]} \right\|^2_{\ell \searrow \ell + 2^i} \\ &\leq \tau \cdot \langle s^{[j+2^k]}, s^{[j+2^k]} \rangle + \tau \cdot \sum_{\ell \in U_i \cap [j, j+2^k)} \left\| x^{[\ell]} \right\|^2_{\ell \searrow \ell + 2^i} \end{aligned}$$

and thus

$$\langle s^{[j+2^k]}, s^{[j+2^k]} \rangle \le \frac{\tau}{1-\tau} \cdot \sum_{\ell \in U_i \cap [j, j+2^k)} \left\| x^{[\ell]} \right\|^2_{\ell \searrow \ell + 2^i}$$
$$\le 1.5\tau \sum_{\ell \in U_i \cap [j, j+2^k)} \left\| x^{[\ell]} \right\|^2_{\ell \searrow \ell + 2^i},$$

since $\tau \le \frac{1}{64}$. Plugging the above into (16), we have

$$\left| \Phi(z^{[j+2^k]}) - \Phi(\widetilde{z}^{[j+2^k]}) \right| \le 2\tau \cdot \|z^{[j]}\|^2_{j \searrow j+2^k} + 3.5\tau \cdot \sum_{\ell \in U_i \cap [j, j+2^k)} \left\| x^{[\ell]} \right\|^2_{\ell \searrow \ell + 2^i}.$$

Therefore, summing up and using (15), we get

$$\|z\|^2_{\mathbf{F}_k} \le \sum_{j \in U_k} \left| \Phi(z^{[j+2^k]}) - \Phi(\widetilde{z}^{[j+2^k]}) \right| \le 2\tau \left(\sum_{j \in U_k} \|z^{[j]}\|^2_{j \searrow j+2^k} \right) + 3.5\tau \left(\sum_{\ell \in U_i} \|x^{[\ell]}\|^2_{\ell \searrow \ell + 2^i} \right)$$
$$= 2\tau \|z\|^2_{\mathbf{F}_k} + 3.5\tau \|x\|^2_{\mathbf{F}_i}.$$

Since $\tau \le \frac{1}{64}$, it follows that

$$\|z\|^2_{\mathbf{F}_k} \le \frac{3.5\tau}{1-2\tau} \cdot \|x\|^2_{\mathbf{F}_i} \le 4\tau \cdot \|x\|^2_{\mathbf{F}_i},$$

completing the proof of Lemma 8.8 in the case $k \ge i$. □

8.3 Bounding $\|\mathbf{L}^{-1}\Delta\mathbf{W}^{(i)}x\|_{\mathbf{F}_k}$ When $i > k$

In the previous subsection, we proved Lemma 8.8 under the assumption that $i \le k$. To handle the case $i > k$, we reduce it to the case $i = k$:

Lemma 8.10. *Under the same setup as Lemma 8.8, for every $i, k \in [\log n]$ such that $k < i$, we have*

$$\left\| \mathbf{L}^{-1}\Delta\mathbf{W}^{(i)}x \right\|^2_{\mathbf{F}_k} = \left\| \mathbf{L}^{-1}\Delta\mathbf{W}^{(i)}x \right\|^2_{\mathbf{F}_i}.$$

Proof. Let $\beta = \Delta\mathbf{W}^{(i)}x$. The key observation here is that $\beta^{[j]}$ is only non-zero for those $j \in U_i$. Letting $z = \mathbf{L}^{-1}\beta$, our goal is to show $\|z\|^2_{\mathbf{F}_k} = \|z\|^2_{\mathbf{F}_i}$.

Since we wish to compute $\|z\|^2_{\mathbf{F}_k}$, we are only interested in $z^{[\ell]}$ for those $\ell \in U_k$. First, we have $z^{[0]} = x^{[0]}$. From the definition $z = \mathbf{L}^{-1}\beta$, for every $j \in U_k$ that $j \ge 2^k$, we have

$$z^{[j]} = \begin{cases} \mathbf{W}_{j \leftarrow (j-2^k)} \cdot z^{[j-2^k]} & j \in U_k \setminus U_i \\ \mathbf{W}_{j \leftarrow (j-2^k)} \cdot z^{[j-2^k]} + \beta^{[j]} & j \in U_i. \end{cases}$$

In particular, for $\ell \in U_k$, let j be the largest element of U_i such that $j \le \ell$. We have

$$z^{[\ell]} = \mathbf{W}_{\ell \leftarrow j} \cdot z^{[j]}. \tag{17}$$

By the definition of $\mathbf{F}_k$ and $\mathbf{F}_i$, we have

$$\|z\|^2_{\mathbf{F}_k} = \sum_{j \in U_k} \left\| z^{[j]} \right\|^2_{j \searrow j+2^k} \qquad \text{and} \qquad \|z\|^2_{\mathbf{F}_i} = \sum_{j \in U_i} \left\| z^{[j]} \right\|^2_{j \searrow j+2^i}.$$

For every $j \in U_i$, by the recursion (17), it holds that

$$\begin{aligned}
\sum_{\ell \in U_k \cap [j, j+2^i)} \|z^{[\ell]}\|^2_{\ell \searrow \ell + 2^k} &= \sum_{\ell \in U_k \cap [j, j+2^i)} \left\| \mathbf{W}_{\ell \leftarrow j} \cdot z^{[j]} \right\|^2_{\ell \searrow \ell + 2^k} \\
&= \sum_{\ell \in U_k \cap [j, j+2^i)} \left\| \mathbf{W}_{\ell \leftarrow j} \cdot z^{[j]} \right\|_2^2 - \left\| \mathbf{W}_{\ell + 2^k \leftarrow j} \cdot z^{[j]} \right\|_2^2 \\
&= \left\| z^{[j]} \right\|_2^2 - \left\| \mathbf{W}_{j+2^i \leftarrow j} \cdot z^{[j]} \right\|_2^2 \qquad \text{(Telescoping sum)} \\
&= \left\| z^{[j]} \right\|^2_{j \searrow j + 2^i}
\end{aligned}$$

Summing up $j \in U_i$ proves the lemma. □

Combining Lemma 8.10 with the proof in the previous subsection completes the proof of Lemma 8.8. Having proven Lemma 8.8, Lemma 8.5 readily follows:

Proof of Lemma 8.5. Fix $k \in [\log n]$. By subadditivity of the norm, we have

$$\begin{aligned}
\left\| \mathbf{L}^{-1} \Delta \mathbf{W} x \right\|_{\mathbf{F}_k} &\leq \sum_{i=1}^{\log n} \left\| \mathbf{L}^{-1} \Delta \mathbf{W}^{(i)} x \right\|_{\mathbf{F}_k} \\
&\leq \sqrt{\log n} \cdot \left(\sum_{i=1}^{\log n} \left\| \mathbf{L}^{-1} \Delta \mathbf{W}^{(i)} x \right\|^2_{\mathbf{F}_k} \right)^{1/2} \\
&\leq \sqrt{\log n} \cdot \left(4\tau \cdot \sum_{i=1}^{\log n} \|x\|^2_{\widetilde{\mathbf{F}}_i} \right)^{1/2}. \qquad \text{(by Lemma 8.8)}
\end{aligned}$$

Hence,

$$\left\| \mathbf{L}^{-1} \Delta \mathbf{W} x \right\|^2_{\mathbf{F}_k} \leq \log n \cdot 4\tau \cdot \|x\|^2_{\widetilde{\mathbf{F}}}.$$

Summing up $k \in [\log(n)]$, we get,

$$\left\| \mathbf{L}^{-1} \Delta \mathbf{W} x \right\|^2_{\mathbf{F}} \leq \log^2 n \cdot 4\tau \cdot \|x\|^2_{\widetilde{\mathbf{F}}}.$$

□

8.4 Applying the Error Reduction Framework

Lemma 8.5 allows us to apply our error reduction framework. Recall from Section 4 that our framework gives us a matrix $\mathbf{A}_m$ such that $\|\mathbf{I} - \mathbf{A}_m \mathbf{L}\|_{\mathbf{F}}$ is low. The following lemma translates this $\mathbf{F}$-seminorm bound into a more useful bound.

Lemma 8.11. *Let B, $\widetilde{\mathcal{W}}$, and τ be as in Lemma 8.5. Let $m \in \mathbb{N}$ and let $\mathbf{A}_m$ be as defined in Section 4. Let $V = V^{(0)} \cup \cdots \cup V^{(n)}$ be the vertices of B. Let $b, y \in \mathbb{R}^V$, where $\mathrm{Supp}(b) \subseteq V^{(0)}$ and $\mathrm{Supp}(y) \subseteq V^{(n)}$. Then*

$$\left| y^T \mathbf{A}_m b - y^T \mathbf{L}^{-1} b \right| \leq (4 \cdot \sqrt{\tau} \cdot \log n)^{(m+1)} \cdot \|y\|_2 \cdot \|b\|_2.$$

Proof. Let $x^* = \mathbf{L}^{-1} b$. By the definition of the $\|\cdot\|_{\mathbf{F}_i}$ norm, we have $\|(\mathbf{A}_m b - x^*)^{[n]}\|_2 \leq \|\mathbf{A}_m b - x^*\|_{\mathbf{F}_i}$ for all $i \in [\log n]$. Consequently,

$$\|(\mathbf{A}_m b - x^*)^{[n]}\|_2 \leq \frac{1}{\sqrt{\log n}} \cdot \|\mathbf{A}_m b - x^*\|_{\mathbf{F}}.$$

By Lemma 8.5, $\|\mathbf{L}^{-1}\Delta\mathbf{W}\|_{\mathbf{F}} \le 2\sqrt{\tau}\cdot\log n \le 1/4$. By Theorem 4.4, it follows that $\|\mathbf{I}-\mathbf{A}_m\mathbf{L}\|_{\mathbf{F}} \le \alpha$, where

$$\alpha = (4\cdot\sqrt{\tau}\cdot\log n)^{(m+1)}.$$

Consequently, $\|x^* - \mathbf{A}_m\mathbf{L}x^*\|_{\mathbf{F}} \le \alpha\cdot\|x^*\|_{\mathbf{F}}$, i.e.,

$$\|\mathbf{A}_m b - x^*\|_{\mathbf{F}} \le \alpha\cdot\|x^*\|_{\mathbf{F}} = \alpha\cdot\sqrt{\sum_{i=1}^{\log n}\|x^*\|_{\mathbf{F}_i}^2}.$$

By a telescoping sum, for every $i\in[\log n]$, we have

$$\|x^*\|_{\mathbf{F}_i}^2 = \sum_{j\in U_i}\left\|\mathbf{W}_{j\leftarrow 0}\cdot b^{[0]}\right\|_{j\searrow j+2^i}^2 = \sum_{j\in U_i}\left[\left\|\mathbf{W}_{j\leftarrow 0}\cdot b^{[0]}\right\|_2^2 - \left\|\mathbf{W}_{j+2^i\leftarrow 0}\cdot b^{[0]}\right\|_2^2\right] \le \left\|b^{[0]}\right\|_2^2 = \|b\|_2^2.$$

Chaining everything together, we get

$$\|(\mathbf{A}_m b - x^*)^{[n]}\|_2 \le \frac{1}{\sqrt{\log n}}\cdot\alpha\cdot\sqrt{\log n}\cdot\|b\|_2 = \alpha\cdot\|b\|_2,$$

and hence $y^T(\mathbf{A}_m b - \mathbf{L}^{-1}b) \le \alpha\cdot\|y\|_2\cdot\|b\|_2$. □

Finally, Theorem 8.3 readily follows from Lemma 8.11:

Proof of Theorem 8.3. Let $b\in\{0,1\}^{(n+1)w}$ be the indicator vector for the start state of B, and let $y\in\{0,1\}^{(n+1)\cdot w}$ be the indicator vector for the accepting states of B. Note that $\|b\|_2 = 1$ and $\|y\|_2 \le \sqrt{w}$. By Lemma 8.11, it suffices to compute $y^T\mathbf{A}_m b$ for $m = \log(\|y\|_2/\varepsilon) \le \log(w/\varepsilon)$, which can be done in $\widetilde{O}(\log(nw)\cdot\log\log(1/\varepsilon))$ space; see Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's work for details [AKMPSV20, Section 7.3]. □

9 Improved WPRG for Unbounded-Width Permutation ROBPs

In this section, we present our improved WPRG for unbounded-width standard-order permutation ROBPs (Theorem 1.9). The WPRG construction is closely related to the non-black-box derandomization algorithm for regular ROBPs that we presented in the previous section.

9.1 PRG with Moderate SV-Error

Recall that the first step of the non-black-box derandomization algorithm was showing that recursively applying a version of the derandomized square operation gives a moderate-error sv-approximation (Theorem 8.1). In the context of *permutation* programs, it is well known [RV05; HPV21] that the derandomized square operation corresponds to the Impagliazzo-Nisan-Wigderson (INW) PRG [INW94]. Consequently, the INW generator fools permutation programs with moderately low "sv-error," as defined below:

Definition 9.1 (Fooling with sv-error)**.** *Let $\mathcal{G}\colon\{0,1\}^s\to\{0,1\}^n$ be a PRG. Let $\mathcal{F}$ be a class of functions $\mathbf{B}\colon\{0,1\}^n\to\{0,1\}^{w\times w}$. We say that $\mathcal{G}$* fools *$\mathcal{F}$* with sv-error *ε if for every $\mathbf{B}\in\mathcal{F}$, if we let $\mathbf{W} = \mathbb{E}[\mathbf{B}]$ and $\widetilde{\mathbf{W}} = \mathbb{E}_{z\in\{0,1\}^s}[\mathbf{B}(\mathcal{G}(z))]$, then $\widetilde{\mathbf{W}} \overset{\text{sv}}{\approx}_\varepsilon \mathbf{W}$.*

Specifically, the INW generator achieves the following parameters:

Theorem 9.2 (PRG for permutation programs with moderate sv-error)**.** *For every $n \in \mathbb{N}$ and $\tau > 0$, there is an explicit PRG $\mathcal{G}\colon \{0,1\}^s \to \{0,1\}^n$ with seed length*

$$s = O\left(\log n \cdot (\log\log n + \log(1/\tau))\right)$$

that fools unbounded-width standard-order permutation ROBPs with sv-error τ.

Like Theorem 8.1, Theorem 9.2 readily follows from the analysis by Ahmadinejad, Peebles, Pyne, Sidford, and Vadhan [APPSV23]. See Appendix B. We remark that Theorem 9.2 strengthens Hoza, Pyne, and Vadhan's analysis of the INW generator [HPV21], because singular-value approximation implies entrywise approximation and "unit-circle" approximation.

9.2 Low-Error High-Seed-Length WPRG: Independent Seeds

To construct our low-error WPRG, we will combine Theorem 9.2 with an error reduction procedure. Fix $\varepsilon > 0$, and fix some PRG $\mathcal{G}\colon \{0,1\}^s \to \{0,1\}^n$ that fools unbounded-width standard-order permutation ROBPs with sv-error τ, where $\varepsilon < \tau < 1/\mathrm{polylog}(n)$. We will show hot convert $\mathcal{G}$ into a WPRG that fools unbounded-width length-n single-accept-state standard-order permutation ROBPs with error ε and seed length

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

To conclude, we will choose $\tau \approx 2^{-\sqrt{\log(1/\varepsilon)}}$.

The first stage of this error reduction procedure corresponds to the error reduction step of our non-black-box derandomization algorithm from Section 8. For our non-black-box derandomization algorithm, we used our general error reduction framework described in Section 4. To construct a WPRG, we instead apply the black-box version of our error reduction framework, described in Section 5. In particular, Section 5 explains how to convert $\mathcal{G}$ into a pseudodistribution $\mathcal{X}^{(m)}$. We now argue that $\mathcal{X}^{(m)}$ fools permutation ROBPs with low entrywise error.

Claim 9.3 ($\mathcal{X}^{(m)}$ fools permutation ROBPs with low entrywise error)**.** *Let $n \in \mathbb{N}$, let $\tau \in (0, \frac{1}{64\log^2 n})$, and let $\mathcal{G}$ be a PRG that fools unbounded-width length-n standard-order permutation ROBPs with sv-error τ. Let $m \in \mathbb{N}$, let $\mathcal{X}^{(m)}$ be the corresponding pseudodistibution constructed in Section 5, and let*

$$\alpha = (4 \cdot \sqrt{\tau} \cdot \log n)^{(m+1)}.$$

Let B be an unbounded-width length-n standard-order permutation ROBP, and let $\mathbf{B}\colon \{0,1\}^n \to \{0,1\}^{w\times w}$ be the corresponding matrix-valued function. Then $\mathcal{X}^{(m)}$ fools $\mathbf{B}$ with entrywise error α, i.e., for every u, v, we have

$$\left|\widetilde{\mathbb{E}}[\mathbf{B}(\mathcal{X}^{(m)})]_{v,u} - \mathbb{E}[\mathbf{B}]_{v,u}\right| \leq \alpha.$$

Proof. Recall from Section 5 that $\mathcal{G}$ and B induce an approximation ensemble $\widetilde{\mathcal{W}} = \{\widetilde{\mathbf{W}}_{j\leftarrow i} : (i,j) \in E(\mathsf{SC})\}$. Because $\mathcal{G}$ fools B with sv-error τ, and more generally each prefix $\mathcal{G}_{i\to j}$ fools the subprogram $B_{j\leftarrow i}$ with sv-error τ, the ensemble $\widetilde{\mathcal{W}}$ is τ-sv-accurate with respect to B. Therefore, we may apply Lemma 8.11. By letting $b, y \in \mathbb{R}^{(n+1)\cdot w}$ be the indicator vectors for the states $(0,u)$ and (n,v) respectively, Lemma 8.11 gives us

$$|(\mathbf{A}_m - \mathbf{L}^{-1})_{(n,v),(0,u)}| \leq \alpha,$$

where $\mathbf{L}$ is the Laplacian matrix of B and $\mathbf{A}_m$ is defined in Section 4. As explained in Section 4, $\mathbb{E}[\mathbf{B}]$ is the $(n,0)$ block of $\mathbf{L}^{-1}$, and as explained in Section 5 (Claim 5.7), $\widetilde{\mathbb{E}}[\mathbf{B}(\mathcal{X}^{(m)})]$ is the $(n,0)$ block of $\mathbf{A}_m$. □

9.3 Final WPRG Construction: Correlated Seeds

Claim 9.3 shows that $\mathcal{X}^{(m)}$ fools unbounded-width single-accept-state permutation ROBPs with low error. Now, similar to Section 6.3, we show how to decrease the seed length of $\mathcal{X}^{(m)}$ by using an additional INW PRG [INW94]. This step is almost exactly the same as our construction and analysis Section 6.3; the main difference is that we invoke Hoza, Pyne, and Vadhan's analysis of the INW PRG [HPV21]. We present further details below.

The final WPRG construction. Just like in Section 6.3, we can write $\mathcal{X}^{(m)}$ in the form

$$\mathcal{X}^{(m)} = \sum_{i=1}^{K} \sigma_i \cdot \sum_{y^{(1)},\dots,y^{(r)} \in \{0,1\}^s} 2^{-sr} \cdot \mathcal{G}_{i,1}(y^{(1)}) \circ \cdots \circ \mathcal{G}_{i,r}(y^{(r)}),$$

where $K = n^{O(m)}$, K is a power of two, $r = O(m \log n)$, $\sigma_i \in \{-1, 0, +1\}$, $\mathcal{G}_{i,j}$ maps $\{0,1\}^s \to \{0,1\}^{n_{i,j}}$ for some $0 \le n_{i,j} \le n$, and $\circ$ denotes string concatenation. This time, let $\mathcal{Y}$ be a distribution over $(\{0,1\}^s)^r$ that γ-fools unbounded-width standard-order *permutation* ROBPs over the alphabet $\{0,1\}^s$ that have a single accepting vertex, where $\gamma = \frac{\varepsilon}{2K}$. Hoza, Pyne, and Vadhan [HPV21] show (using the INW construction [INW94]) that we can explicitly sample $\mathcal{Y}$ using a seed of length

$$q = O(s + (\log(1/\gamma) + \log\log r) \cdot \log r) = O(s + \log(1/\gamma) \cdot \log r).$$

Just like in Section 6.3, write $\mathcal{Y}$ using pseudodistribution notation as

$$\mathcal{Y} = \sum_{z \in \{0,1\}^q} 2^{-q} \cdot y_z^{(1)} \circ \cdots \circ y_z^{(r)},$$

where $y_z^{(j)} \in \{0,1\}^s$ for each $j \in [r]$. Our final pseudodistribution $\mathcal{Z}$ over $\{0,1\}^n$ is given by the formula

$$\mathcal{Z} = \sum_{i=1}^{K} \sigma_i \cdot \sum_{z \in \{0,1\}^q} 2^{-q} \cdot \mathcal{G}_{i,1}(y_z^{(1)}) \circ \cdots \circ \mathcal{G}_{i,r}(y_z^{(r)}).$$

Error and seed length. Let B be an unbounded-width length-n standard-order permutation ROBP with a single accepting vertex. Just like in Section 6.3, one can show that

$$\left| \mathbb{E}[B(\mathcal{Z})] - \mathbb{E}[B(\mathcal{X}^{(m)})] \right| \le \varepsilon/2.$$

We will choose $\tau < 1/(64 \cdot \log^3 n)$ and $m = \Theta(\frac{\log(1/\varepsilon)}{\log(1/\tau)})$, so the error α in Claim 9.3 is at most $\varepsilon/2$. This ensures that $\mathcal{Z}$ fools B with error ε.

The pseudodistribution $\mathcal{Z}$ corresponds to a WPRG with seed length

$$\begin{aligned}
\log K + q &= O(s + \log K + \log(1/\gamma) \cdot \log r) \\
&= O(s + (m \log n + \log(1/\varepsilon)) \cdot (\log m + \log\log n)) \\
&\le O\left(s + \left(\frac{\log n \cdot \log(1/\varepsilon)}{\log(1/\tau)} + \log(1/\varepsilon)\right) \cdot \log\log(n/\varepsilon)\right).
\end{aligned}$$

We take $\mathcal{G}$ to be the PRG from Theorem 9.2, so $s = O(\log n \cdot (\log(1/\tau) + \log\log n))$. Furthermore, we choose

$$\tau = \min\left\{2^{-\sqrt{\log(1/\varepsilon)\cdot\log\log(n/\varepsilon)}}, \frac{1}{64\log^3 n}\right\}.$$

Therefore, the overall seed length becomes

$$O\left(\log n \cdot \sqrt{\log(1/\varepsilon)\cdot\log\log(n/\varepsilon)} + \log(1/\varepsilon)\cdot\log\log(n/\varepsilon)\right),$$

completing the proof of Theorem 1.9.

Acknowledgments

We want to thank Edward Pyne and Salil Vadhan for helpful discussions.

References

[AKMPSV20] A. Ahmadinejad, J. A. Kelner, J. Murtagh, J. Peebles, A. Sidford, and S. 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*. Ed. by S. Irani. IEEE, 2020, pp. 1295–1306 (cit. on pp. 5, 6, 7, 9, 11, 12, 17, 19, 20, 23, 54, 55, 61, 68, 77).

[APPSV23] A. Ahmadinejad, J. Peebles, E. Pyne, A. Sidford, and S. P. Vadhan. "Singular Value Approximation and Reducing Directed to Undirected Graph Sparsification". In: *CoRR* abs/2301.13541 (2023) (cit. on pp. 20, 54, 62, 68, 73).

[AKS87] M. Ajtai, J. Komlós, and E. Szemerédi. "Deterministic Simulation in LOGSPACE". In: *Proceedings of the 19th Symposium on Theory of Computing (STOC)*. 1987, pp. 132–140 (cit. on p. 2).

[BDVY13] A. Bogdanov, Z. Dvir, E. Verbin, and A. Yehudayoff. "Pseudorandomness for width-2 branching programs". In: *Theory Comput.* 9 (2013), pp. 283–292 (cit. on p. 3).

[BHPP22] A. Bogdanov, W. M. Hoza, G. Prakriya, and E. Pyne. "Hitting Sets for Regular Branching Programs". In: *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*. Vol. 234. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 3:1–3:22 (cit. on pp. 4, 5, 6, 9).

[BCG20] M. Braverman, G. Cohen, and S. Garg. "Pseudorandom pseudo-distributions with near-optimal error for read-once branching programs". In: *SIAM J. Comput.* 49.5 (2020), STOC18-242–STOC18-299 (cit. on pp. 2, 3, 33).

[BRRY14] M. Braverman, A. Rao, R. Raz, and A. Yehudayoff. "Pseudorandom generators for regular branching programs". In: *SIAM Journal on Computing* 43.3 (2014), pp. 973–986 (cit. on pp. 4, 5, 9, 11, 14, 15, 16, 17, 18, 19, 33, 34, 35, 38, 43, 53).

[BV10] J. Brody and E. Verbin. "The Coin Problem and Pseudorandomness for Branching Programs". In: *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2010, pp. 30–39 (cit. on p. 4).

[CHHL19] E. Chattopadhyay, P. Hatami, K. Hosseini, and S. Lovett. "Pseudorandom generators from polarizing random walks". In: *Theory Comput.* 15 (2019), Paper No. 10, 26 (cit. on p. 4).

[CL20] E. Chattopadhyay and J.-J. Liao. "Optimal Error Pseudodistributions for Read-Once Branching Programs". In: *35th Computational Complexity Conference (CCC 2020)*. Ed. by S. Saraf. Vol. 169. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, 25:1–25:27 (cit. on pp. 2, 3, 33).

[CLTW23] L. Chen, X. Lyu, A. Tal, and H. Wu. "New PRGs for Unbounded-Width/Adaptive-Order Read-Once Branching Programs". In: *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*. Ed. by K. Etessami, U. Feige, and G. Puppis. Vol. 261. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 39:1–39:20 (cit. on pp. 5, 6).

[CH22] K. Cheng and W. M. Hoza. "Hitting sets give two-sided derandomization of small space". In: *Theory Comput.* 18 (2022), Paper No. 21, 32 (cit. on pp. 2, 6).

[CDRST21] G. Cohen, D. Doron, O. Renard, O. Sberlo, and A. Ta-Shma. "Error Reduction for Weighted PRGs Against Read Once Branching Programs". In: *36th Computational Complexity Conference (CCC 2021)*. Ed. by V. Kabanets. Vol. 200. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 22:1–22:17 (cit. on pp. 2, 3, 6, 9, 10, 11, 15, 29, 30, 33, 34, 37).

[CDST22] G. Cohen, D. Doron, O. Sberlo, and A. Ta-Shma. "Approximating Iterated Multiplication of Stochastic Matrices in Small Space". In: *Electron. Colloquium Comput. Complex.* TR22-149 (2022) (cit. on pp. 6, 9).

[CKKPPRS18] M. B. Cohen, J. A. Kelner, R. Kyng, J. Peebles, R. Peng, A. B. Rao, and A. Sidford. "Solving Directed Laplacian Systems in Nearly-Linear Time through Sparse LU Factorizations". In: *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*. IEEE Computer Society, 2018, pp. 898–909 (cit. on p. 6).

[CKPPRSV17] M. B. Cohen, J. A. Kelner, J. Peebles, R. Peng, A. B. Rao, A. Sidford, and A. Vladu. "Almost-linear-time algorithms for Markov chains and new spectral primitives for directed graphs". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*. ACM, 2017, pp. 410–419 (cit. on p. 6).

[CKPPSV16] M. B. Cohen, J. A. Kelner, J. Peebles, R. Peng, A. Sidford, and A. Vladu. "Faster Algorithms for Computing the Stationary Distribution, Simulating Random Walks, and More". In: *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*. IEEE Computer Society, 2016, pp. 583–592 (cit. on p. 6).

[De11] A. De. "Pseudorandomness for permutation and regular branching programs". In: *26th Annual IEEE Conference on Computational Complexity*. IEEE Computer Soc., Los Alamitos, CA, 2011, pp. 221–231 (cit. on pp. 4, 6).

[DMRTV21] D. Doron, R. Meka, O. Reingold, A. Tal, and S. Vadhan. "Pseudorandom Generators for Read-Once Monotone Branching Programs". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)*. Ed. by M. Wootters and L. Sanità. Vol. 207. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 58:1–58:21 (cit. on p. 43).

[FK18] M. A. Forbes and Z. Kelley. "Pseudorandom Generators for Read-Once Branching Programs, in Any Order". In: *FOCS*. IEEE Computer Society, 2018, pp. 946–955 (cit. on pp. 40, 50).

[GG81] O. Gabber and Z. Galil. "Explicit constructions of linear-sized superconcentrators". In: *J. Comput. System Sci.* 22.3 (1981). Special issued dedicated to Michael Machtey, pp. 407–420 (cit. on p. 76).

[Gol11] O. Goldreich. "Basic facts about expander graphs". In: *Studies in complexity and cryptography*. Vol. 6650. Lecture Notes in Comput. Sci. Springer, Heidelberg, 2011, pp. 451–464 (cit. on p. 76).

[GV22] L. Golowich and S. Vadhan. "Pseudorandomness of expander random walks for symmetric functions and permutation branching programs". In: *37th Computational Complexity Conference (CCC 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik. 2022 (cit. on p. 4).

[GMRTV12] P. Gopalan, R. Meka, O. Reingold, L. Trevisan, and S. Vadhan. "Better pseudorandom generators from milder pseudorandom restrictions". In: *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science—FOCS 2012*. IEEE Computer Soc., Los Alamitos, CA, 2012, pp. 120–129 (cit. on p. 3).

[HH23] P. Hatami and W. M. Hoza. *Theory of Unconditional Pseudorandom Generators*. ECCC preprint TR23-019. 2023 (cit. on p. 3).

[HPV21] W. M. Hoza, E. Pyne, and S. Vadhan. "Pseudorandom generators for unbounded-width permutation branching programs". In: *12th Innovations in Theoretical Computer Science (ITCS'21)* (2021) (cit. on pp. 2, 4, 5, 6, 23, 61, 62, 63).

[Hoz21] W. M. Hoza. "Better Pseudodistributions and Derandomization for Space-Bounded Computation". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*. Vol. 207. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 28:1–28:23 (cit. on pp. 2, 3, 6, 9, 11, 29, 30).

[Hoz22] W. M. Hoza. "Recent Progress on Derandomizing Space-Bounded Computation". In: *Bulletin of EATCS* 138.3 (2022) (cit. on p. 2).

[HZ20] W. M. Hoza and D. Zuckerman. "Simple optimal hitting sets for small-success **RL**". In: *SIAM J. Comput.* 49.4 (2020), pp. 811–820 (cit. on p. 9).

[INW94] R. Impagliazzo, N. Nisan, and A. Wigderson. "Pseudorandomness for network algorithms". In: *STOC*. ACM, 1994, pp. 356–364 (cit. on pp. 6, 15, 23, 29, 37, 61, 63, 69, 78).

[KNP11] M. Koucký, P. Nimbhorkar, and P. Pudlák. "Pseudorandom generators for group products [extended abstract]". In: *STOC'11—Proceedings of the 43rd ACM Symposium on Theory of Computing*. ACM, New York, 2011, pp. 263–272 (cit. on p. 4).

[LPV22] C. H. Lee, E. Pyne, and S. Vadhan. "Fourier growth of regular branching programs". In: *Approximation, randomization, and combinatorial optimization. Algorithms and techniques*. Vol. 245. LIPIcs. Leibniz Int. Proc. Inform. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2022, Art. No. 2, 21 (cit. on pp. 4, 6).

[LPV23] C. H. Lee, E. Pyne, and S. Vadhan. "On the Power of Regular and Permutation Branching Programs". In: *Electronic Colloquium on Computational Complexity (ECCC)* (2023) (cit. on p. 4).

[Mar73] G. A. Margulis. "Explicit constructions of expanders". In: *Problemy Peredači Informacii* 9.4 (1973), pp. 71–80 (cit. on p. 76).

[MRT19] R. Meka, O. Reingold, and A. Tal. "Pseudorandom generators for width-3 branching programs". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*. ACM, 2019, pp. 626–637 (cit. on pp. 3, 15, 16, 39, 43, 44, 49, 50).

[Mih89] M. Mihail. "Conductance and convergence of Markov chains-a combinatorial treatment of expanders". In: *30th Annual Symposium on Foundations of Computer Science*. 1989, pp. 526–531 (cit. on p. 70).

[MRSV21] J. Murtagh, O. Reingold, A. Sidford, and S. P. Vadhan. "Derandomization beyond Connectivity: Undirected Laplacian Systems in Nearly Logarithmic Space". In: *SIAM J. Comput.* 50.6 (2021), pp. 1892–1922 (cit. on p. 70).

[NN93] J. Naor and M. Naor. "Small-bias probability spaces: efficient constructions and applications". In: *SIAM J. Comput.* 22.4 (1993), pp. 838–856 (cit. on p. 50).

[Nis92] N. Nisan. "Pseudorandom generators for space-bounded computation". In: *Combinatorica* 12.4 (1992), pp. 449–461 (cit. on p. 1).

[PP22] A. L. Putterman and E. Pyne. "Near-Optimal Derandomization of Medium-Width Branching Programs". In: *Electron. Colloquium Comput. Complex.* TR22-150 (2022) (cit. on pp. 6, 9).

[PRZ23] E. Pyne, R. Raz, and W. Zhan. *Certified Hardness vs. Randomness for Log-Space*. ECCC preprint TR23-040. 2023 (cit. on p. 2).

[PV22] E. Pyne and S. Vadhan. "Deterministic approximation of random walks via queries in graphs of unbounded size". In: *5th SIAM Symposium on Simplicity in Algorithms*. [Society for Industrial and Applied Mathematics (SIAM)], Philadelphia, PA, 2022, pp. 57–67 (cit. on pp. 4, 6).

[PV21] E. Pyne and S. P. Vadhan. "Pseudodistributions That Beat All Pseudorandom Generators (Extended Abstract)". In: *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*. Ed. by V. Kabanets. Vol. 200. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 33:1–33:15 (cit. on pp. 2, 3, 4, 5, 6, 9, 10, 15, 23, 29, 30, 33, 34, 37).

[RSV13] O. Reingold, T. Steinke, and S. Vadhan. "Pseudorandomness for regular branching programs via Fourier analysis". In: *Approximation, randomization, and combinatorial optimization*. Vol. 8096. Lecture Notes in Comput. Sci. Springer, Heidelberg, 2013, pp. 655–670 (cit. on p. 4).

[RTV06] O. Reingold, L. Trevisan, and S. Vadhan. "Pseudorandom walks on regular digraphs and the **RL** vs. **L** problem". In: *STOC'06: Proceedings of the 38th Annual ACM Symposium on Theory of Computing*. ACM, New York, 2006, pp. 457–466 (cit. on p. 4).

[RVW02] O. Reingold, S. Vadhan, and A. Wigderson. "Entropy waves, the zig-zag graph product, and new constant-degree expanders". In: *Annals of mathematics* (2002), pp. 157–187 (cit. on p. 68).

[RV05] E. Rozenman and S. Vadhan. "Derandomized squaring of graphs". In: *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2005, pp. 436–447 (cit. on pp. 11, 19, 20, 23, 24, 54, 61, 68, 69, 77, 78, 79).

[SZ95] M. Saks and D. Zuckerman. Unpublished work showing that ε-biased distributions fool width-2 ROBPs with error $O(\varepsilon \cdot n)$. 1995 (cit. on p. 3).

[SZ99] M. E. Saks and S. Zhou. "BP $_{\mathrm{H}}$Space(S) subseteq DSPACE($S^{3/2}$)". In: *J. Comput. Syst. Sci.* 58.2 (1999), pp. 376–403 (cit. on p. 3).

[Ste12] T. Steinke. *Pseudorandomness for Permutation Branching Programs Without the Group Theory*. ECCC preprint TR12-083. 2012 (cit. on p. 4).

[SVW17] T. Steinke, S. Vadhan, and A. Wan. "Pseudorandomness and Fourier-growth bounds for width-3 branching programs". In: *Theory Comput.* 13 (2017), Paper No. 12, 50 (cit. on p. 3).

A Derandomized Products and Singular-Value Approximation

In this section, we prove Theorem 8.1. In fact, we will prove the more general version below.

Theorem A.1 (General version of Theorem 8.1)**.** *There is a deterministic algorithm that, given a regular width-w length-n standard-order ROBP B and a value $\tau \in (0,1)$, outputs a matrix $\widetilde{\mathbf{W}}_{n\leftarrow 0} \in \mathbb{R}^{w\times w}$ such that $\widetilde{\mathbf{W}}_{j\leftarrow 0} \overset{\mathrm{sv}}{\approx}_{\tau} \mathbf{W}_{n\leftarrow 0}$, where $\mathbf{W}_{n\leftarrow 0}$ is defined from B as in Section 4. The algorithm uses $\widetilde{O}(\log(nw)\cdot\log(1/\tau))$ bits of space.*

The algorithm will essentially be a repeated application of Rozenman and Vadhan's derandomized squaring operation [RV05]. (We assume that n is a power of two; this is without loss of generality because we can always pad B with identity transitions without changing its functionality.) This is essentially the same as the base case algorithm used by Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan [AKMPSV20, Section 7.3]. We reiterate that the fact that the iterated derandomized square computes a moderate-error sv-approximation already follows from the work of Ahmadinejad, Peebles, Pyne, Sidford, and Vadhan [APPSV23]; we include our proof primarily for the sake of having a relatively self-contained presentation.

A.1 Labelings, Rotation Maps, and Derandomized Products

To describe the algorithm, we first need to recall (variations of) some standard definitions from the literature [RVW02; RV05].

Definition A.2 (Regular bigraph)**.** *A* bigraph *is a triple $G = (U, V, E)$, where U and V are sets of vertices and E is a set of directed edges going from U to V. The* transition matrix *of G is the matrix $\mathbf{W} \in \mathbb{R}^{V \times U}$ in which $\mathbf{W}_{v,u}$ is the fraction of outgoing edges from u that lead to v. We say that G is d-*regular *if each vertex $u \in U$ has precisely d outgoing edges and each vertex $v \in V$ has precisely d incoming edges. (In this case, $\mathbf{W}$ is doubly stochastic.)*

Definition A.3 (One-way labeling)**.** *A* one-way labeling *of a d-regular bigraph $G = (U, V, E)$ assigns each edge $e \in E$ a label in $[d]$ such that for every $u \in U$, the outgoing edges of u have distinct labels. If G has a one-way labeling, then we write $G[u, i] = v$ when there is an edge (u, v) with label i.*

Definition A.4 (Two-way labeling)**.** *A* two-way labeling *of a d-regular bigraph $G = (U, V, E)$ is a labeling of the edges in G such that*

1. *Every edge (u, v) has two labels in $[d]$, an "outgoing label" and an "incoming label."*
2. *For every vertex $u \in U$, the outgoing labels of the outgoing edges of u are distinct.*
3. *For every vertex $v \in V$, the incoming labels of the incoming edges of v are distinct.*

A two-way labeling induces a one-way labeling by considering only the outgoing labels.

Definition A.5 (Rotation map)**.** *Let $G = (U, V, E)$ be a d-regular bigraph with a two-way labeling. The* rotation map $\mathrm{Rot}_G \colon U \times [d] \to V \times [d]$ *is defined as follows:* $\mathrm{Rot}_G(u, i) = (v, j)$ *if there is an edge $(u, v) \in E$ with outgoing label i and incoming label j. Note that in this case, we have $G[u, i] = v$.*

As mentioned previously, our algorithm is based on the "derandomized square" construction by Rozenman and Vadhan [RV05]. We slightly generalize it to a "derandomized product" construction.

Definition A.6 (Derandomized Product)**.** *Let $G_1 = (U, V, E_1)$ and $G_2 = (V, W, E_2)$ be d-regular bigraphs, where G_1 has a two-way labeling and G_2 has a one-way labeling. Let $H = ([d], [d], E_H)$ be a c-regular bigraph with a one-way labeling. The* derandomized product $G_2 \textcircled{p}_H G_1 = (U, W, E)$ *is $(c \cdot d)$-regular bigraph with a one-way labeling defined as follows. To compute $(G_2 \textcircled{p}_H G_1)[v_0, (i_0, j_0)]$ where $v_0 \in U$ and $(i_0, j_0) \in [d] \times [c]$:*

1. *Let $(v_1, i_1) = \mathrm{Rot}_{G_1}(v_0, i_0)$.*
2. *Let $i_2 = H[i_1, j_0]$.*
3. *Let $v_2 = G_2[v_1, i_2]$.*
4. *Output $(G_2 \textcircled{p}_H G_1)[v_0, (i_0, j_0)] := v_2$.*

Remark A.7 (Incoming edge labels for $G_2 \textcircled{p}_H G_1$)**.** *Ultimately, we wish to apply Definition A.6 recursively. However, Definition A.6 requires that G_1 has a two-way labeling, whereas it merely defines a one-way labeling for $G_2 \textcircled{p}_H G_1$. If we wish to apply the operation multiple times, say $G_3 \textcircled{p}_{H'} (G_2 \textcircled{p}_H G_1)$, we must first assign incoming edge labels to $G_2 \textcircled{p}_H G_1$.*

Following prior work going back to Rozenman and Vadhan's paper [RV05], we will study two distinct methods *of assigning incoming edge labels to $G_2 \textcircled{p}_H G_1$. One method (explained in Section A.4) works in the general setting of regular ROBPs, but unfortunately it leads to non-black-box algorithms. The other method (explained in Appendix B) can be interpreted as a black-box algorithm – indeed, it is essentially equivalent to the Impagliazzo-Nisan-Wigderson PRG [INW94] – but unfortunately it only works in the special case of permutation ROBPs.*

Before explaining either of these methods of assigning incoming edge labels, we first analyze the derandomized product operation in a way that is agnostic *about the issue of incoming edge labels.*

Remark A.8. *Murtagh, Reingold, Sidford, and Vadhan also defined a "derandomized product" operation and used the notation* Ⓟ *[MRSV21]. Our definition is similar but not identical to theirs.*

A.2 The Derandomized Product SV-Approximates the Exact Product

Recall that the derandomized product $G_2 \,Ⓟ_H\, G_1$ is parameterized by an auxiliary graph H. We will take H to be a *spectral expander*, a concept that we review in the two definitions below.

Definition A.9 (Complete bigraph)**.** *For each $d \in \mathbb{N}$, let $K_{d,d} = ([d], [d], [d] \times [d])$ denote the complete bigraph, and let $\mathbf{J}_{d\times d}$ denote the transition matrix of $K_{d,d}$. That is, $\mathbf{J}_{d\times d}$ is a $d \times d$ matrix, and every entry of $\mathbf{J}_{d\times d}$ is $1/d$. When the dimension d is clear from context, we simply write $\mathbf{J}$.*

Definition A.10 (Spectral expansion [Mih89])**.** *Let $H = (U, V, E_H)$ be a regular bigraph with transition matrix $\mathbf{H} \in \mathbb{R}^{V\times U}$, where $|U| = |V|$. We define*

$$\lambda(H) = \|\mathbf{J} - \mathbf{H}\|_2.$$

The spectral expansion *of H is the quantity $1 - \lambda(H)$.*

(Definition A.10 considers the general case of a c-regular bigraph, but our expanders will always be induced by c-regular *undirected* graphs that are expanders in the standard spectral sense.)

Definition A.11 (Parallel and perpendicular components)**.** *Let 1_d denote the d-dimensional vector in which every coordinate is 1. When the dimension d is clear from context, we simply write 1. For a vector $z \in \mathbb{R}^d$, let $z_\parallel$ and $z_\perp$ denote the projections of z onto the subspace parallel and perpendicular to 1_d, respectively. Formally, we define $z_\parallel = \mathbf{J} \cdot z$ and $z_\perp = z - z_\parallel$.*

Note that if H is a regular bigraph with transition matrix $\mathbf{H}$, then $\mathbf{H} \cdot 1 = 1$, so

$$\|(\mathbf{J} - \mathbf{H}) \cdot z\|_2 \le \lambda(H) \cdot \|z_\perp\|_2.$$

We will use the following additional pieces of notation to analyze the derandomized product.

Definition A.12 (Transpose bigraph)**.** *If $G = (U, V, E)$ is a bigraph, then the* transpose bigraph *G^T is defined by reversing the direction of each edge $e \in E$. If G has a two-way labeling, then we define a two-way labeling on G^T by the rule*

$$\mathrm{Rot}_{G^T} \equiv (\mathrm{Rot}_G)^{-1}.$$

Definition A.13 (Restrictions with respect to v)**.** *Let $G_1 = (U, V, E_1)$ and $G_2 = (V, W, E_2)$ be d-regular bigraphs. Assume that G_1 has a two-way labeling and G_2 has a one-way labeling. Let $x \in \mathbb{R}^U$ and $y \in \mathbb{R}^W$ be vectors over the vertices in U and W respectively. For each $v \in V$, we define the* restriction of y with respect to v*, denoted $y^{(v)} \in \mathbb{R}^d$, to be the subvector of y indexed by out-neighbors of v, i.e.,*

$$y^{(v)}_i = y_{G_2[v,i]} \quad \text{for every } i \in [d].$$

Similarly, we define the restriction of x with respect to v*, denoted $x^{(v)} \in \mathbb{R}^d$, to be the subvector of x indexed by in-neighbors of v, i.e.,*

$$x^{(v)}_i = x_{G_1^T[v,i]} \quad \text{for every } i \in [d].$$

Remark A.14. *For convenience, we write $x_{\|}^{(v)}$ to denote the parallel component of the restriction of x with respect to v, i.e., $x_{\|}^{(v)} = (x^{(v)})_{\|}$. Note that in general, this vector (the parallel component of the restriction) is different than $(x_{\|})^{(v)}$ (the restriction of the parallel component). Similarly, we omit parentheses in the expressions $x_{\perp}^{(v)}$, $y_{\|}^{(v)}$, and $y_{\perp}^{(v)}$.*

The following lemma is the key to analyzing the derandomized product operation.

Lemma A.15 (Approximation property of derandomized product)**.** *Let $G_1 = (U, V, E_1)$ and $G_2 = (V, W, E_2)$ be d-regular bigraphs, where G_1 has a two-way labeling and G_2 has a one-way labeling. Let $H = ([d], [d], E_H)$ be a regular bigraph with a one-way labeling. Let $\mathbf{W}_1$ and $\mathbf{W}_2$ be the transition matrices of G_1 and G_2 respectively, and let $\widetilde{\mathbf{W}}$ be the transition matrix of $G_2 \textcircled{p}_H G_1$. Then for every $x \in \mathbb{R}^U$ and $y \in R^W$, we have*

$$\left| y^T (\mathbf{W}_2 \mathbf{W}_1 - \widetilde{\mathbf{W}}) x \right| \leq \frac{\lambda(H)}{d} \cdot \sum_{v \in V} \left\| x_{\perp}^{(v)} \right\|_2 \cdot \left\| y_{\perp}^{(v)} \right\|_2 .$$

Proof. Let $\mathbf{H}$ be the transition matrix of H. By Definition A.6, we have

$$y^T \widetilde{\mathbf{W}} x = \frac{1}{d} \cdot \sum_{v \in V} \sum_{i_1 \in [d]} \sum_{i_2 \in [d]} y_{G_2[v,i_2]} \cdot \mathbf{H}_{i_2,i_1} \cdot x_{G_1^T[v,i_1]},$$

i.e., we sum $y_w \cdot \mathbf{H}_{i_2,i_1} \cdot x_u$ over all length-two paths (u, v, w), where i_1 and i_2 are the labels of the incoming and outgoing edges of v in the path. By Definition A.13, this can be written as

$$y^T \widetilde{\mathbf{W}} x = \frac{1}{d} \sum_{v \in V} (y^{(v)})^T \cdot \mathbf{H} \cdot x^{(v)}. \tag{18}$$

Observe that $\mathbf{W}_2 \mathbf{W}_1$ is the transition matrix of $G_2 \textcircled{p}_{K_{d,d}} G_1$. Therefore, as a special case of (18), we have

$$y^T \mathbf{W}_2 \mathbf{W}_1 x = \frac{1}{d} \sum_{v \in V} (y^{(v)})^T \cdot \mathbf{J}_{d \times d} \cdot x^{(v)}.$$

Therefore, by the spectral expansion property, we have

$$|y^T (\mathbf{W}_2 \mathbf{W}_1 - \widetilde{\mathbf{W}}) x| = \frac{1}{d} \left| \sum_v (y^{(v)})^T \cdot (\mathbf{J}_{d \times d} - \mathbf{H}) \cdot x^{(v)} \right| \leq \frac{\lambda(H)}{d} \cdot \sum_v \left\| y_{\perp}^{(v)} \right\|_2 \cdot \left\| x_{\perp}^{(v)} \right\|_2 ,$$

as desired. □

Using Lemma A.15, we can now prove that the derandomized product singular-value approximates the exact product:

Corollary A.16 (Derandomized product sv-approximates exact product)**.** *Under the assumptions of Lemma A.15, we have $\widetilde{\mathbf{W}} \overset{\text{sv}}{\approx}_{2\lambda(H)} \mathbf{W}_2 \mathbf{W}_1$.*

Proof. Let $\mathbf{W}_{21} = \mathbf{W}_2 \mathbf{W}_1$. Let $x \in \mathbb{R}^V$ and $y \in \mathbb{R}^W$. By Lemma A.15, we have

$$\begin{aligned} |y^T (\mathbf{W}_{21} - \widetilde{\mathbf{W}}) x| &\leq \frac{\lambda(H)}{d} \sum_v \left\| y_{\perp}^{(v)} \right\|_2 \left\| x_{\perp}^{(v)} \right\|_2 \\ &\leq \frac{\lambda(H)}{d} \sqrt{\sum_v \left\| y_{\perp}^{(v)} \right\|_2^2} \cdot \sqrt{\sum_v \left\| x_{\perp}^{(v)} \right\|_2^2}. \end{aligned} \tag{19}$$

We can rewrite

$$\sum_v \left\|y_\perp^{(v)}\right\|_2^2 = \sum_v \left(\left\|y^{(v)}\right\|_2^2 - \left\|y_\parallel^{(v)}\right\|_2^2\right)$$
$$\sum_v \left\|x_\perp^{(v)}\right\|_2^2 = \sum_v \left(\left\|x^{(v)}\right\|_2^2 - \left\|x_\parallel^{(v)}\right\|_2^2\right).$$

We first calculate $\sum_v \|y^{(v)}\|^2$. By Definition A.13, each $y_w, w \in W$ contributes to exactly d entries among $y_i^{(v)}$ ($v \in V$ and $i \in [d]$). Also, each $y_i^{(v)} = y_w$ for some $w \in W$. Therefore,

$$\sum_v \left\|y^{(v)}\right\|_2^2 = \sum_v \sum_i (y_i^{(v)})^2 = d \cdot \sum_w y_w^2 = d \cdot \|y\|_2^2.$$

Next, we calculate $\|y_\parallel^{(v)}\|_2^2$. Observe that $y_\parallel^{(v)} = 1_d \cdot c_v$ where $c_v = (y^T \mathbf{W}_2)_v$. Hence,

$$\sum_v \left\|y_\parallel^{(v)}\right\|_2^2 = \sum_v d \cdot |(y^T \mathbf{W}_2)_v|^2 = d \cdot y^T \mathbf{W}_2 \mathbf{W}_2^T y,$$

and

$$\sum_v \left\|y_\perp^{(v)}\right\|_2^2 = d \cdot \left(\|y\|_2^2 - y^T \mathbf{W}_2 \mathbf{W}_2^T y\right) = d \cdot \|y\|_{\mathbf{I}-\mathbf{W}_2\mathbf{W}_2^T}^2.$$

Symmetrically,

$$\sum_v \left\|x^{(v)}\right\|_2^2 = d \cdot \|x\|_2^2,$$
$$\sum_v \left\|x_\parallel^{(v)}\right\|_2^2 = \sum_v d \cdot |(\mathbf{W}_1 x)_v|^2 = d \cdot x^T \mathbf{W}_1^T \mathbf{W}_1 x,$$

and

$$\sum_v \left\|x_\perp^{(v)}\right\|_2^2 = d \cdot \left(\|x\|_2^2 - x^T \mathbf{W}_1^T \mathbf{W}_1 x\right) = d \cdot \|x\|_{\mathbf{I}-\mathbf{W}_1^T\mathbf{W}_1}^2.$$

Therefore, by the AM-GM inequality,

$$|y^T(\mathbf{W}_{21} - \widetilde{\mathbf{W}})x| \le \lambda(H) \cdot \|x\|_{\mathbf{I}-\mathbf{W}_1^T\mathbf{W}_1} \cdot \|y\|_{\mathbf{I}-\mathbf{W}_2\mathbf{W}_2^T} \le \frac{\lambda(H)}{2} \cdot \left(\|x\|_{\mathbf{I}-\mathbf{W}_1^T\mathbf{W}_1}^2 + \|y\|_{\mathbf{I}-\mathbf{W}_2\mathbf{W}_2^T}^2\right).$$

Since $\mathbf{W}_2$ is doubly stochastic, we have $\|\mathbf{W}_{21}x\|_2 \le \|\mathbf{W}_1 x\|_2$, and therefore

$$\|x\|_{\mathbf{I}-\mathbf{W}_1^T\mathbf{W}_1} \le \|x\|_{\mathbf{I}-\mathbf{W}_{21}^T\mathbf{W}_{21}}.$$

Similarly, since $\mathbf{W}_1$ is doubly stochastic, we have $\|\mathbf{W}_{21}^T y\|_2 \le \|\mathbf{W}_2^T y\|_2$, and therefore

$$\|y\|_{\mathbf{I}-\mathbf{W}_2\mathbf{W}_2^T} \le \|y\|_{\mathbf{I}-\mathbf{W}_{21}\mathbf{W}_{21}^T}.$$

Therefore,

$$|y^T(\mathbf{W}_{21} - \widetilde{\mathbf{W}})x| \le \frac{\lambda(H)}{2} \cdot \left(\|x\|_{\mathbf{I}-\mathbf{W}_{21}^T\mathbf{W}_{21}}^2 + \|y\|_{\mathbf{I}-\mathbf{W}_{21}\mathbf{W}_{21}^T}^2\right). \qquad \square$$

A.3 Recursive Derandomized Product: Accumulation of Error

We have shown that a single application of the derandomized product operation has bounded error in the sense of sv-approximation. To analyze multiple applications of the derandomized product operation, we must establish a few facts about sv-approximation. The proofs are relatively short and only use elementary linear algebra. The first lemma allows us to bound $\|(\widetilde{\mathbf{W}} - \mathbf{W})x\|_2$ in terms of $\|x\|_{\mathbf{I}-\mathbf{W}^T\mathbf{W}}$ assuming $\widetilde{\mathbf{W}} \overset{\text{sv}}{\approx} \mathbf{W}$.

Lemma A.17. *Let $\mathbf{W}, \widetilde{\mathbf{W}}$ be doubly stochastic $w \times w$ matrices, and suppose $\widetilde{\mathbf{W}} \overset{\text{sv}}{\approx}_\varepsilon \mathbf{W}$. Then, for all $x, y \in \mathbb{R}^w$, we have*

$$\|(\widetilde{\mathbf{W}} - \mathbf{W})x\|_2 \le \frac{\varepsilon}{2} \cdot \|x\|_{\mathbf{I}-\mathbf{W}^T\mathbf{W}} \qquad \text{and} \qquad \|(\widetilde{\mathbf{W}} - \mathbf{W})^T y\|_2 \le \frac{\varepsilon}{2} \cdot \|y\|_{\mathbf{I}-\mathbf{W}\mathbf{W}^T}.$$

Proof. We prove the claim for the x side. The y side follows by taking transposes. Let

$$\eta := \frac{\|(\widetilde{\mathbf{W}} - \mathbf{W})x\|_2}{\|x\|_{\mathbf{I}-\mathbf{W}^T\mathbf{W}}}.$$

We have

$$\begin{aligned}
\|(\widetilde{\mathbf{W}} - \mathbf{W})x\|_2^2 &= x^T(\widetilde{\mathbf{W}} - \mathbf{W})^T(\widetilde{\mathbf{W}} - \mathbf{W})x \\
&= \left(\tfrac{1}{\sqrt{\eta}}x\right)^T (\widetilde{\mathbf{W}} - \mathbf{W})^T(\widetilde{\mathbf{W}} - \mathbf{W})(\sqrt{\eta}x) \\
&\le \frac{\varepsilon}{4}\left(\|\sqrt{\eta}x\|_{\mathbf{I}-\mathbf{W}^T\mathbf{W}}^2 + \left\|(\widetilde{\mathbf{W}} - \mathbf{W}) \cdot \frac{x}{\sqrt{\eta}}\right\|_{\mathbf{I}-\mathbf{W}\mathbf{W}^T}^2\right) \\
&= \frac{\varepsilon}{4}\left(\eta \cdot \|x\|_{\mathbf{I}-\mathbf{W}^T\mathbf{W}}^2 + \frac{1}{\eta}\|(\widetilde{\mathbf{W}} - \mathbf{W})x\|_{\mathbf{I}-\mathbf{W}\mathbf{W}^T}^2\right) \\
&\le \frac{\varepsilon}{4}\left(\eta \cdot \|x\|_{\mathbf{I}-\mathbf{W}^T\mathbf{W}}^2 + \frac{1}{\eta}\|(\widetilde{\mathbf{W}} - \mathbf{W})x\|_2^2\right) \qquad \text{(since } \|\cdot\|_{\mathbf{I}-\mathbf{W}\mathbf{W}^T} \le \|\cdot\|_2\text{)} \\
&\le \frac{\varepsilon}{2} \cdot \|(\widetilde{\mathbf{W}} - \mathbf{W})x\|_2 \cdot \|x\|_{\mathbf{I}-\mathbf{W}^T\mathbf{W}}.
\end{aligned}$$

Re-arranging proves that $\|(\widetilde{\mathbf{W}} - \mathbf{W})x\|_2 \le (\varepsilon/2) \cdot \|x\|_{\mathbf{I}-\mathbf{W}^T\mathbf{W}}$ as desired. □

The next lemma shows that if $\widetilde{\mathbf{W}}_1 \overset{\text{sv}}{\approx} \mathbf{W}_1$ and $\widetilde{\mathbf{W}}_2 \overset{\text{sv}}{\approx} \mathbf{W}_2$, then $\widetilde{\mathbf{W}}_2 \cdot \widetilde{\mathbf{W}}_1 \overset{\text{sv}}{\approx} \mathbf{W}_2 \cdot \mathbf{W}_1$. Note that versions of this lemma and the next can also be found in the recent work by Ahmadinejad, Peebles, Pyne, Sidford, and Vadhan [APPSV23]. Still, we include proofs here in the interest of having a self-contained presentation. Also, our proofs appear to be more elementary.

Lemma A.18 (Product of sv-approximations is sv-approximation of product)**.** *Let $\mathbf{W}_1, \mathbf{W}_2, \widetilde{\mathbf{W}}_1, \widetilde{\mathbf{W}}_2$ be doubly stochastic $w \times w$ matrices. Assume that $\widetilde{\mathbf{W}}_1 \overset{\text{sv}}{\approx}_\varepsilon \mathbf{W}_1$ and $\widetilde{\mathbf{W}}_2 \overset{\text{sv}}{\approx}_\varepsilon \mathbf{W}_2$. Then*

$$\widetilde{\mathbf{W}}_2 \cdot \widetilde{\mathbf{W}}_1 \overset{\text{sv}}{\approx}_{\varepsilon\cdot(1+\varepsilon/2)} \mathbf{W}_2 \cdot \mathbf{W}_1.$$

Proof. For convenience, write $\mathbf{W}_{21} = \mathbf{W}_2 \cdot \mathbf{W}_1$. Let $x, y \in \mathbb{R}^w$. We use the following decomposition:

$$y^T(\mathbf{W}_{21} - \widetilde{\mathbf{W}}_2\widetilde{\mathbf{W}}_1)x = \underbrace{y^T(\mathbf{W}_2 - \widetilde{\mathbf{W}}_2)\mathbf{W}_1 x}_{(*)} + \underbrace{y^T\mathbf{W}_2(\mathbf{W}_1 - \widetilde{\mathbf{W}}_1)x}_{(**)} - \underbrace{y^T(\mathbf{W}_2 - \widetilde{\mathbf{W}}_2)(\mathbf{W}_1 - \widetilde{\mathbf{W}}_1)x}_{(***)}.$$

We bound the three terms separately. First,

$$|(*)| = |y^T(\mathbf{W}_2 - \widetilde{\mathbf{W}}_2)\mathbf{W}_1 x| \le \frac{\varepsilon}{4}\left(\|y\|^2_{\mathbf{I}-\mathbf{W}_2\mathbf{W}_2^T} + \|\mathbf{W}_1 x\|^2_{\mathbf{I}-\mathbf{W}_2^T\mathbf{W}_2}\right).$$

Symmetrically,

$$|(**)| \le \frac{\varepsilon}{4}\left(\|\mathbf{W}_2^T y\|^2_{\mathbf{I}-\mathbf{W}_1\mathbf{W}_1^T} + \|x\|^2_{\mathbf{I}-\mathbf{W}_1^T\mathbf{W}_1}\right).$$

Next, we bound (***) as

$$\begin{aligned}
|(***)| &= |y^T(\mathbf{W}_2 - \widetilde{\mathbf{W}}_2)(\mathbf{W}_1 - \widetilde{\mathbf{W}}_1)x| \\
&\le \|(\mathbf{W}_2 - \widetilde{\mathbf{W}}_2)^T y\|_2 \cdot \|(\mathbf{W}_1 - \widetilde{\mathbf{W}}_1)x\|_2 \\
&\le \frac{\varepsilon^2}{4}\|y\|_{\mathbf{I}-\mathbf{W}_2\mathbf{W}_2^T} \cdot \|x\|_{\mathbf{I}-\mathbf{W}_1^T\mathbf{W}_1} \qquad \text{(Lemma A.17)} \\
&\le \frac{\varepsilon^2}{8}\left(\|x\|^2_{\mathbf{I}-\mathbf{W}_1^T\mathbf{W}_1} + \|y\|^2_{\mathbf{I}-\mathbf{W}_2\mathbf{W}_2^T}\right) \\
&\le \frac{\varepsilon^2}{8}\left(\|x\|^2_{\mathbf{I}-\mathbf{W}_{21}^T\mathbf{W}_{21}} + \|y\|^2_{\mathbf{I}-\mathbf{W}_{21}\mathbf{W}_{21}^T}\right).
\end{aligned}$$

We finish the proof by utilizing the following equations:

$$\begin{aligned}
\|x\|^2_{\mathbf{I}-\mathbf{W}_1^T\mathbf{W}_1} + \|\mathbf{W}_1 x\|^2_{\mathbf{I}-\mathbf{W}_2^T\mathbf{W}_2} &= \|x\|^2_{\mathbf{I}-\mathbf{W}_{21}^T\mathbf{W}_{21}}, \\
\|y\|^2_{\mathbf{I}-\mathbf{W}_2\mathbf{W}_2^T} + \|\mathbf{W}_2^T y\|^2_{\mathbf{I}-\mathbf{W}_1\mathbf{W}_1^T} &= \|y\|^2_{\mathbf{I}-\mathbf{W}_{21}\mathbf{W}_{21}^T},
\end{aligned}$$

and adding up the three bounds on quantities (*), (**), and (***). □

The next lemma shows that singular-value approximation is transitive (with some modest loss in the approximation parameter).

Lemma A.19 (Transitivity of sv-approximation)**.** *Let $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{R}^{w\times w}$ be three doubly stochastic matrices. Suppose $\mathbf{A} \overset{\text{sv}}{\approx}_{\varepsilon_1} \mathbf{B}$ and $\mathbf{B} \overset{\text{sv}}{\approx}_{\varepsilon_2} \mathbf{C}$ where $\varepsilon_2 \in [0, 4]$. Then $\mathbf{A} \overset{\text{sv}}{\approx}_{\varepsilon} \mathbf{C}$, where $\varepsilon = \varepsilon_1 + \varepsilon_2 + 2\varepsilon_1\varepsilon_2$. In particular, if $\varepsilon_2 \in [0, 1/2]$, then $\varepsilon \le 2\varepsilon_1 + \varepsilon_2$.*

Proof. Let $x, y \in \mathbb{R}^w$ be arbitrary. Our job is to show that

$$|y^T(\mathbf{A} - \mathbf{C})x| \le \frac{\varepsilon}{4}\left(\|x\|^2_{\mathbf{I}-\mathbf{C}^T\mathbf{C}} + \|y\|^2_{\mathbf{I}-\mathbf{C}\mathbf{C}^T}\right). \tag{20}$$

To show this, we first use the triangle inequality and get

$$|y^T(\mathbf{A} - \mathbf{C})x| \le |y^T(\mathbf{A} - \mathbf{B})x| + |y^T(\mathbf{B} - \mathbf{C})x|.$$

We bound the right-hand side now. The second term is easy:

$$|y^T(\mathbf{B} - \mathbf{C})x| \le \frac{\varepsilon_2}{4}\left(\|x\|^2_{\mathbf{I}-\mathbf{C}^T\mathbf{C}} + \|y\|^2_{\mathbf{I}-\mathbf{C}\mathbf{C}^T}\right).$$

For the first term, we start with the inequality

$$\begin{aligned}
|y^T(\mathbf{A} - \mathbf{B})x| &\le \frac{\varepsilon_1}{4}\left(\|x\|^2_{\mathbf{I}-\mathbf{B}^T\mathbf{B}} + \|y\|^2_{\mathbf{I}-\mathbf{B}\mathbf{B}^T}\right) \\
&= \frac{\varepsilon_1}{4}\left(\|x\|^2_{\mathbf{I}-\mathbf{C}^T\mathbf{C}} + \|y\|^2_{\mathbf{I}-\mathbf{C}\mathbf{C}^T}\right) + \frac{\varepsilon_1}{4}\left(\|\mathbf{C}x\|^2 - \|\mathbf{B}x\|^2 + \|\mathbf{C}^T y\|^2 - \|\mathbf{B}^T y\|^2\right).
\end{aligned}$$

Now, it is clear that, to show (20), it suffices to show the following inequalities.

$$\|\mathbf{C}x\|^2 - \|\mathbf{B}x\|^2 \le 2\varepsilon_2 \|x\|^2_{\mathbf{I}-\mathbf{C}^T\mathbf{C}}, \qquad \text{and}$$
$$\|\mathbf{C}^T y\|^2 - \|\mathbf{B}^T y\|^2 \le 2\varepsilon_2 \|y\|^2_{\mathbf{I}-\mathbf{C}\mathbf{C}^T}.$$

In the following, we show the proof of the first inequality. The second one can be verified via a symmetric argument. We observe that

$$\begin{aligned}
\|\mathbf{C}x\|^2 - \|\mathbf{B}x\|^2 &= x^T(\mathbf{C}^T\mathbf{C} - \mathbf{B}^T\mathbf{B})x \\
&= x^T\left((\mathbf{C}^T - \mathbf{B}^T)\mathbf{C} + \mathbf{C}^T(\mathbf{C} - \mathbf{B}) - (\mathbf{C}^T - \mathbf{B}^T)(\mathbf{C} - \mathbf{B})\right)x \\
&\le 2 \cdot |x^T\mathbf{C}^T(\mathbf{C} - \mathbf{B})x| + |x^T(\mathbf{C}^T - \mathbf{B}^T)(\mathbf{C} - \mathbf{B})x| \\
&\le \frac{\varepsilon_2}{2}\left(\|x\|^2_{\mathbf{I}-\mathbf{C}^T\mathbf{C}} + \|\mathbf{C}x\|^2_{\mathbf{I}-\mathbf{C}\mathbf{C}^T}\right) + \frac{\varepsilon_2^2}{4}\|x\|^2_{\mathbf{I}-\mathbf{C}^T\mathbf{C}} && \text{(Lemma A.17)} \\
&\le (\varepsilon_2 + \varepsilon_2^2/4)\|x\|^2_{\mathbf{I}-\mathbf{C}^T\mathbf{C}} && (\|\mathbf{C}x\|^2_{\mathbf{I}-\mathbf{C}\mathbf{C}^T} \le \|x\|^2_{\mathbf{I}-\mathbf{C}^T\mathbf{C}}, \text{ by Fact 8.7}) \\
&\le 2\varepsilon_2\|x\|^2_{\mathbf{I}-\mathbf{C}^T\mathbf{C}}, && \text{(Assumption on } \varepsilon_2)
\end{aligned}$$

as desired. □

Now we are ready to analyze the recursive derandomized product. We will prove the following bound.

Lemma A.20 (Recursive derandomized product sv-approximates the exact product)**.** *Let $n, d, c \in \mathbb{N}$. For each $t \in \{0, 1, \ldots, \log n\}$ and each $j \in U_t \setminus \{n\}$, let $\widetilde{G}_{j+2^t \leftarrow j} = (V^{(j)}, V^{(j+2^t)}, E^{(j+2^t \leftarrow j)})$ be a $(d \cdot c^t)$-regular bigraph with a two-way labeling. Furthermore, let $\lambda \in (0, \frac{1}{6\log^2 n})$, and for each $t \in [\log n]$, let $H_t = ([d \cdot c^{t-1}], [d \cdot c^{t-1}], E_{H_t})$ be a c-regular bigraph with a one-way labeling satisfying $\lambda(H_t) \le \lambda$. Assume also that for each $t \in [\log n]$ and each $j \in U_t \setminus \{n\}$, we have*

$$\widetilde{G}_{j+2^t \leftarrow j} = \widetilde{G}_{j+2^t \leftarrow j+2^{t-1}} \textcircled{p}_{H_t} \widetilde{G}_{j+2^{t-1} \leftarrow j}$$

(where the equation above merely denotes equality as graphs with one-way *labelings). For each $(i, j) \in E(\mathsf{SC}_n)$, let $\widetilde{\mathbf{W}}_{j \leftarrow i}$ be the transition matrix of $\widetilde{G}_{j \leftarrow i}$, and let $\mathbf{W}_{j \leftarrow i} = \widetilde{\mathbf{W}}_{j \leftarrow j-1} \cdot \widetilde{\mathbf{W}}_{j-1 \leftarrow j-2} \cdots \widetilde{\mathbf{W}}_{i+1 \leftarrow i}$. Then*

$$\widetilde{\mathbf{W}}_{n \leftarrow 0} \overset{\text{sv}}{\approx}_{11\lambda \cdot \log n} \mathbf{W}_{n \leftarrow 0}.$$

To be clear, the conclusion of Lemma A.20 holds *regardless* of how the incoming edge labels are defined in $\widetilde{G}_{j+2^t \leftarrow j}$.

Proof. We will indeed prove the following stronger claim, from which the lemma above will immediately follow.

Claim A.21. *For every $t \in [\log n]$ and $j \in U_t \setminus \{n\}$, we have $\widetilde{\mathbf{W}}_{j+2^t \leftarrow j} \overset{\text{sv}}{\approx}_{\varepsilon_t} \mathbf{W}_{j+2^t \leftarrow j}$, where*

$$\varepsilon_t = (1 + 6\lambda \cdot \log n)^{t-1} \cdot t \cdot 4\lambda.$$

Proof. We prove the claim by induction on t. For the case $t = 1$, this claim is saying for every $j \in U_1 \setminus \{n\}$, we have

$$\widetilde{\mathbf{W}}_{j+2 \leftarrow j} \overset{\text{sv}}{\approx}_{4\lambda} \mathbf{W}_{j+2 \leftarrow j},$$

which is true by Corollary A.16.

Suppose the claim is true for $t-1$. We verify the case of $t > 1$. Fix $j \in U_t \setminus \{n\}$. By Corollary A.16, we have

$$\widetilde{\mathbf{W}}_{j+2^t \leftarrow j} \overset{\text{sv}}{\approx}_{2\lambda} \widetilde{\mathbf{W}}_{j+2^t \leftarrow j+2^{t-1}} \cdot \widetilde{\mathbf{W}}_{j+2^{t-1} \leftarrow j}. \tag{21}$$

By the induction hypothesis, we obtain

$$\widetilde{\mathbf{W}}_{j+2^t \leftarrow j+2^{t-1}} \overset{\text{sv}}{\approx}_{\varepsilon_{t-1}} \mathbf{W}_{j+2^t \leftarrow j+2^{t-1}},$$
$$\widetilde{\mathbf{W}}_{j+2^{t-1} \leftarrow j} \overset{\text{sv}}{\approx}_{\varepsilon_{t-1}} \mathbf{W}_{j+2^{t-1} \leftarrow j}.$$

By Lemma A.18, we get

$$\widetilde{\mathbf{W}}_{j+2^t \leftarrow j+2^{t-1}} \cdot \widetilde{\mathbf{W}}_{j+2^{t-1} \leftarrow j} \overset{\text{sv}}{\approx}_{\varepsilon_{t-1} \cdot (1+\varepsilon_{t-1}/2)} \mathbf{W}_{j+2^t \leftarrow j}. \tag{22}$$

By our assumption $\lambda \leq 1/(6\log^2 n)$ and the fact that $t \leq \log n$, we have

$$1 + \varepsilon_{t-1}/2 = 1 + (1 + 6\lambda \cdot \log n)^{t-2} \cdot t \cdot 2\lambda \leq 1 + e \cdot t \cdot 2\lambda \leq 1 + 6\lambda \cdot \log n.$$

Therefore,

$$\varepsilon_{t-1} \cdot (1 + \varepsilon_{t-1}/2) \leq (1 + 6\lambda \cdot \log n)^{t-1} \cdot (t-1) \cdot 4\lambda \leq 1/2,$$

and furthermore,

$$4\lambda + \varepsilon_{t-1} \cdot (1 + \varepsilon_{t-1}/2) \leq \varepsilon_t.$$

By Lemma A.19 applied to (21) and (22), it follows that $\widetilde{\mathbf{W}}_{j+2^t \leftarrow j} \overset{\text{sv}}{\approx}_{\varepsilon_t} \mathbf{W}_{j+2^t \leftarrow j}$, which completes the induction step, and the proof of the claim. □

Finally, since $\lambda \in \left(0, \frac{1}{6\log^2 n}\right)$, it follows that $(1 + 6\lambda \cdot \log n)^{\log n - 1} \cdot \log n \cdot 4\lambda < 4e \cdot \lambda \cdot \log n \leq 11\lambda \cdot \log n$. This completes the proof of the lemma. □

A.4 The Algorithm: Assigning Incoming Edge Labels

Having completed our analysis of the derandomized product operation, we now discuss "implementing" the recursive derandomized product and thereby proving Theorem A.1. For the auxiliary graph H, we will use the following family of space-efficient expanders:

Lemma A.22 (Space-efficient expanders)**.** *For every $d \in \mathbb{N}$ that is a power of two, for every $\lambda \in (0,1)$, there is a bigraph $H = ([d], [d], E_H)$ with a two-way labeling satisfying the following.*

- $\lambda(H) \leq \lambda$.
- *H is c-regular where c is a power of two and $c \leq \text{poly}(1/\lambda)$.*
- *Rot_H can be evaluated in space that is linear in its input length, i.e., space $O(\log(d/\lambda))$.*

Proof sketch. We begin by constructing a c_0-regular (undirected) graph H_0, with transition matrix $\mathbf{H}_0$, such that $c_0 = O(1)$, c_0 is a power of two, and $\|\mathbf{J} - \mathbf{H}_0\|_2 \leq 1 - \Omega(1)$:

- If d is a power of four, then d is a perfect square, so we can let H_0 be the Margulis-Gabber-Galil graph [Mar73; GG81], which can be taken to have degree $c_0 = 8$ [Gol11].
- If d is not a power of four, then $d/2$ is a power of four. Therefore, we take H_0 to be the tensor product of the Margulis-Gabber-Galil graph with the complete graph on two vertices (with self loops), so we get $c_0 = 16$.

Next, we let H_0' be the $O(\log(1/\lambda))$-th power of H_0. Finally, to construct H, we include two directed edges (u,v) and (v,u) for each undirected edge $\{u,v\}$ in H_0'. □

Let us now describe the algorithm of Theorem A.1. Recall that we are given a regular width-w length-n standard-order ROBP B and a value $\delta_0 \in (0,1)$. Let $V^{(0)},\dots,V^{(n)}$ be the vertex layers of B, and let $E^{(1)},\dots,E^{(n)}$ be the edge layers, so $E^{(i)} \subseteq V^{(i-1)} \times V^{(i)}$. For each $j \in \{0,\dots,n-1\}$, let $\widetilde{G}_{j+1\leftarrow j} = (V^{(j)}, V^{(j+1)}, E^{(j+1)})$, so $\widetilde{G}_{j+1\leftarrow j}$ is a d-regular bigraph with a one-way labeling where $d = 2$. In this initial stage, we *arbitrarily* assign distinct incoming labels to the incoming edges of each vertex $v \in V^{(j+1)}$, so $\widetilde{G}_{j+1\leftarrow j}$ has a two-way labeling.

Our plan is to recursively multiply these bigraphs $\widetilde{G}_{j+1\leftarrow j}$ using the derandomized product operation. Define

$$\lambda = \min\left\{\frac{\tau}{11\log n}, \frac{1}{6\log^2 n}\right\},$$

and for each $t \in [\log n]$, let $H_t = ([d\cdot c^{i-1}], [d\cdot c^{i-1}], E_{H_t})$ be the c-regular bigraph with $\lambda(H_t) \le \lambda$ from Lemma A.22. Fix $(j, j+2^t) \in E(\mathsf{SC}_n)$ where $t \ge 1$. Let $G_1 = \widetilde{G}_{j+2^{t-1}\leftarrow j}$, let $G_2 = \widetilde{G}_{j+2^t\leftarrow j+2^{t-1}}$, and let $H = H_t$. By induction, we have already defined G_1 and G_2 to be $(d\cdot c^{t-1})$-regular bigraphs with two-way labelings. Roughly speaking, we wish to set

$$\widetilde{G}_{j+2^t\leftarrow j} = G_2 \circledcirc_H G_1. \tag{23}$$

However, the derandomized product operation only assigns outgoing edge labels, and we need $\widetilde{G}_{j+2^t\leftarrow j}$ to have a two-way labeling (for future rounds of the inductive process), so we need to assign incoming edge labels. Following prior work [RV05; AKMPSV20], we therefore define $\widetilde{G}_{j+2^t\leftarrow j}$ by the following rotation map. To compute $\mathrm{Rot}_{\widetilde{G}_{j+2^t\leftarrow j}}(v_0,(i_0,j_0))$:

1. Let $(v_1,i_1) = \mathrm{Rot}_{G_1}(v_0,i_0)$.
2. Let $(i_2,j_1) = \mathrm{Rot}_{H}(i_1,j_0)$.
3. Let $(v_2,i_3) = \mathrm{Rot}_{G_2}(v_1,i_2)$.
4. Output $\mathrm{Rot}_{\widetilde{G}_{j+2^t\leftarrow j}}(v_0,(i_0,j_0)) := (v_2,(i_3,j_1))$.

Observe that this definition is compatible with the derandomized product definition, i.e., it ensures that (23) holds as an equality of graphs with one-way labelings.

Claim A.23 (Space efficiency of recursive derandomized product)**.** *Let $\widetilde{\mathbf{W}}_{j\leftarrow i}$ be the transition matrix of the graph $\widetilde{G}_{j\leftarrow i}$ defined above. Given B and τ, the matrix $\widetilde{\mathbf{W}}_{n\leftarrow 0}$ can be computed in space*

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

Since our derandomized graph product is slightly different than the formalisms in prior work, we include a sketch of the proof of Claim A.23, but we stress that there is no real novelty here; the proof is essentially the same as the corresponding proofs in prior work.

Proof sketch. Let $v \in V^{(0)}$ and $e = (e_0,e_1,\dots,e_{\log n}) \in [d]\times[c]^{\log n}$. Given v and e, one can compute $\mathrm{Rot}_{\widetilde{G}_{n\leftarrow 0}}(v,e)$ by the following algorithm, which essentially consists of "unrolling" the recursive definition of $\widetilde{G}_{n\leftarrow 0}$:

1. For $i = 1$ to n:

(a) Update $(v, e_0) \leftarrow \mathrm{Rot}_{\widetilde{G}_{i \leftarrow i-1}}(v, e_0)$, so now $v \in V^{(i)}$.

(b) If $i < n$:

i. Let $t \in [\log n]$ be the smallest positive integer such that i is not a multiple of 2^t, i.e., the binary expansion of i has precisely $t-1$ trailing zeroes.

ii. Update $(e_0, \ldots, e_{t-1}) \leftarrow \mathrm{Rot}_{H_t}((e_0, \ldots, e_{t-1}), e_t)$.

2. Output (v, e).

The algorithm above uses $O(\log w + \log n \cdot \log c)$ bits of space for storing i, v, e, etc., doing arithmetic, and computing $\mathrm{Rot}_{\widetilde{G}_{i \leftarrow i-1}}$, plus $O(\log n \cdot \log c)$ additional bits for computing Rot_{H_t}. Since $c = \mathrm{poly}(1/\lambda) = \mathrm{poly}((\log n)/\tau)$, the total space complexity is $O(\log w + \log n \cdot \log(1/\tau) + \log n \cdot \log\log n)$. Finally, to compute the (v, u) entry of the matrix $\widetilde{\mathbf{W}}_{n \leftarrow 0}$, we compute $\mathrm{Rot}_{\widetilde{G}_{n \leftarrow 0}}(u, e)$ for every e, and count how many times it outputs a pair of the form (v, e'). □

Combining Claim A.23 and Lemma A.20 completes the proof of Theorem A.1, because $11\lambda \cdot \log n \le \tau$.

B The INW Generator

In this section, we prove Theorem 9.2, which states that there is an explicit PRG that fools permutation ROBPs with sv-error τ and seed length $\widetilde{O}(\log n \cdot \log(1/\tau))$. Briefly, the PRG is the famous Impagliazzo-Nisan-Wigderson (INW) PRG [INW94]. Rozenman and Vadhan observed that the INW generator is closely related to their derandomized squaring operation [RV05]. Because of this connection, we will be able to apply the analysis in Appendix A to finish the proof.

In more detail, let us begin by reviewing the construction of the INW generator.

Definition B.1 (The INW Pseudorandom Generator [INW94])**.** *Let n and c be powers of two, let $d = 2$, and for each $t \in [\log n]$, let H_t be a c-regular bigraph $H_t = ([d \cdot c^{t-1}], [d \cdot c^{t-1}], E_{H_t})$ with a one-way labeling. Relative to this family $(H_1, \ldots, H_{\log n})$, we recursively define the INW generator as follows.*

Define $\mathsf{INW}_0 : \{0,1\}^1 \to \{0,1\}^1$ as the trivial PRG, namely $\mathsf{INW}_0(x) = x$. For each $t \in [\log n]$, having defined $\mathsf{INW}_{t-1} : \{0,1\}^{1+(t-1)\cdot\log c} \to \{0,1\}^{2^{t-1}}$, we define $\mathsf{INW}_t : \{0,1\}^{1+t\cdot\log c} \to \{0,1\}^{2^t}$ as $\mathsf{INW}_t(x, y) = \mathsf{INW}_{t-1}(x) \circ \mathsf{INW}_{t-1}(H_t[x, y])$, where $\circ$ denotes string concatenation.

The INW generator is connected to the derandomized product operation via the notion of a *consistent one-way labeling*, defined next.

Definition B.2 (Consistent one-way labeling)**.** *Let $G = (U, V, E)$ be a d-regular bigraph. A* consistent one-way labeling *for G is a one-way labeling such that for each $v \in V$, the incoming edges of v have distinct labels. That is, $G[u, i] = G[v, i]$ implies $u = v$. In this case, we can extend G to a graph $\overline{G}$ with a two-way labeling defined by*

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

i.e., the incoming label of an edge is equal to its outgoing label.

The derandomized product operation preserves the property of having a consistent one-way labeling:

Lemma B.3 ([RV05])**.** *Let $G_1 = (U, V, E_1)$ and $G_2 = (V, W, E_2)$ be d-regular bigraphs with consistent one-way labelings. Let $H = ([d], [d], E_H)$ be a c-regular bigraph with a one-way labeling. Then $G_2 \textcircled{p}_H \overline{G_1}$ has a consistent one-way labeling.*

Proof. Let $u_0, u_0' \in U$, let $i_0 \in [d]$, and let $j_0 \in [c]$. Suppose that $(G_2 \textcircled{p}_H \overline{G_1})[u_0, (i_0, j_0)] = (G_2 \textcircled{p}_H \overline{G_1})[u_0', (i_0, j_0)]$. By the definitions of $\overline{G_1}$ and the derandomized product, this implies that

$$G_2[u_1, i_2] = G_2[u_1', i_2],$$

where $u_1 = G_1[u_0, i_0]$, $u_1' = G_1[u_0', i_0]$, and $i_2 = H[i_0, j_0]$. Because G_2 has a consistent one-way labeling, it follows that $u_1' = u_1$, i.e., $G_1[u_0, i_0] = G_1[u_0', u_0]$. Because G_1 has a consistent one-way labeling, it follows that $u_0 = u_0'$. □

Let $n, c, d, H_1, \ldots, H_{\log n}, \mathsf{INW}_0, \ldots, \mathsf{INW}_{\log n}$ be as in Definition B.1. Let B be a width-w length-n standard-order permutation ROBP with vertex layers $V^{(0)}, \ldots, V^{(n)}$ and edge layers $E^{(1)}, \ldots, E^{(n)}$. For $j \in \{0, \ldots, n-1\}$, let $\widetilde{G}_{j+1 \leftarrow j} = (V^{(j)}, V^{(j+1)}, E^{(j+1)})$, so $\widetilde{G}_{j+1 \leftarrow j}$ is a 2-regular bigraph.

Because B is a *permutation* ROBP, each such graph $\widetilde{G}_{j+1 \leftarrow j}$ has a *consistent* one-way labeling. This fact enables us to recursively multiply these graphs via the derandomized product as follows. Let $t \in [\log n]$ and let $(j, j + 2^t) \in E(\mathsf{SC}_n)$. Let $G_1 = \widetilde{G}_{j+2^{t-1} \leftarrow j}$, let $\widetilde{G}_{j+2^t \leftarrow j+2^{t-1}}$, and let $H = H_t$. By induction, the graphs G_1 and G_2 have consistent one-way labelings. Define

$$\widetilde{G}_{j+2^t \leftarrow j} = G_2 \textcircled{p}_H \overline{G_1},$$

and observe that $\widetilde{G}_{j+2^t \leftarrow j}$ has a consistent one-way labeling by Lemma B.3.

The construction above "corresponds to" the INW generator in the following sense.

Lemma B.4 (INW generator $\leftrightarrow$ derandomized product)**.** *Assume the setup described above. Let $t \in \{0, 1, \ldots, \log n\}$, let $j \in U_t \setminus \{n\}$, let $u \in V^{(j)}$, and let $x \in \{0,1\}^{1+t \cdot \log c}$. Then*

$$\widetilde{G}_{j+2^t \leftarrow j}[u, x] = B[u, \mathsf{INW}_t(x)].$$

Proof. We use induction on t. The case of $t = 0$ follows by definition. For $t \geq 1$, recall that $\mathsf{INW}_t(x, y) = \mathsf{INW}_{t-1}(x) \circ \mathsf{INW}_{t-1}(H[x, y])$, where $H = H_t$. Therefore, the vertex $B[u, \mathsf{INW}_t(x, y)]$ can be computed as follows:

1. Let $v = B[u, \mathsf{INW}_{t-1}(x)]$.
2. Let $z = H[x, y]$.
3. Output $B[v, \mathsf{INW}_{t-1}(z)]$.

Let $G_1 = \widetilde{G}_{j+2^{t-1} \leftarrow j}$ and $G_2 = \widetilde{G}_{j+2^t \leftarrow j+2^{t-1}}$. By the induction hypothesis, the procedure above is equivalent to the following:

1. Let $v = G_1[u, x]$.
2. Let $z = H[x, y]$.
3. Output $G_2[v, z]$.

This is precisely the same as the procedure to compute $(G_2 \textcircled{p}_H \overline{G_1})[u, (x, y)]$. □

Combining Lemma B.4 with our analysis in Appendix A yields Theorem 9.2:

Proof of Theorem 9.2. Let $\lambda = \min\{\tau/(11\log n), 1/(6\log^2 n)\}$. Let $d = 2$, and for each $t \in [\log n]$, let $H_t = ([d \cdot c^{t-1}], [d \cdot c^{t-1}], E_{H_t})$ be the c-regular bigraph with $\lambda(H_t) \leq \lambda$ from Lemma A.22. Let $\mathsf{INW}_0, \ldots, \mathsf{INW}_{\log n}$ be the corresponding INW generators, and let $\mathcal{G} = \mathsf{INW}_{\log n}$, so $\mathcal{G}$ has seed length

$$O(\log n \cdot \log c) = O(\log n \cdot (\log(1/\tau) + \log\log n)).$$

Let B be an unbounded-width length-n standard-order permutation ROBP, and let $\widetilde{G}_{j+2^t \leftarrow j}$ be the corresponding graphs as defined above Lemma B.4. Let $V^{(0)}, \ldots, V^{(n)}$ be the layers of B. By Lemma B.4, for every $u \in V^{(0)}$ and every seed x,

$$B[u, \mathcal{G}(x)] = \widetilde{G}_{n \leftarrow 0}[u, x].$$

Therefore, if we let $\mathbf{B}\colon \{0,1\}^n \to \{0,1\}^{w \times w}$ be the matrix-valued function corresponding to B (namely $\mathbf{B}(x)_{v,u} = B^{v \leftarrow u}(x)$), then $\mathbb{E}[\mathbf{B}] = \mathbf{W}_{n \leftarrow 0}$ and $\mathbb{E}_x[\mathbf{B}(\mathcal{G}(x))] = \widetilde{\mathbf{W}}_{n \leftarrow 0}$, where $\widetilde{\mathbf{W}}_{j \leftarrow i}$ is the transition matrix of $\widetilde{G}_{j \leftarrow i}$ and $\mathbf{W}_{n \leftarrow 0} = \widetilde{\mathbf{W}}_{n \leftarrow n-1} \cdot \widetilde{\mathbf{W}}_{n-1 \leftarrow n-2} \cdots \widetilde{\mathbf{W}}_{1 \leftarrow 0}$. By Lemma A.20, it follows that $\mathbb{E}_x[\mathbf{B}(\mathcal{G}(x))] \overset{\text{sv}}{\approx}_\tau \mathbb{E}[\mathbf{B}]$, and hence $\mathcal{G}$ fools $\mathbf{B}$ with sv-error τ. □

C Product of Singular-Value Approximations

In this section, we prove Lemma 8.6, which is restated below.

Lemma C.1 (Lemma 8.6, restated). *Let $\mathbf{W}_1, \mathbf{W}_2, \widetilde{\mathbf{W}}_1, \widetilde{\mathbf{W}}_2, \widetilde{\mathbf{W}}$ be doubly stochastic matrices. Let $\mathbf{W} = \mathbf{W}_2\mathbf{W}_1$. Assume that $\widetilde{\mathbf{W}}_1 \overset{\text{sv}}{\approx}_\varepsilon \mathbf{W}_1$, $\widetilde{\mathbf{W}}_2 \overset{\text{sv}}{\approx}_\varepsilon \mathbf{W}_2$, and $\widetilde{\mathbf{W}} \overset{\text{sv}}{\approx}_\varepsilon \mathbf{W}$. Then*

$$\mathbf{W}_2\mathbf{W}_1 + (\widetilde{\mathbf{W}} - \widetilde{\mathbf{W}}_2\widetilde{\mathbf{W}}_1) \overset{\text{sv}}{\approx}_{2\varepsilon(1+\varepsilon/4)} \mathbf{W}_2\mathbf{W}_1.$$

Proof. By Lemma A.18, we have $\widetilde{\mathbf{W}}_2\widetilde{\mathbf{W}}_1 \overset{\text{sv}}{\approx}_{\varepsilon(1+\varepsilon/2)} \mathbf{W}_2\mathbf{W}_1$. Therefore, for any vectors x and y, we have

$$\begin{aligned}
|x^T(\widetilde{\mathbf{W}} - \widetilde{\mathbf{W}}_2\widetilde{\mathbf{W}}_1)y| &\leq |x^T(\widetilde{\mathbf{W}} - \mathbf{W}_1\mathbf{W}_2)y| + |x^T(\widetilde{\mathbf{W}}_2\widetilde{\mathbf{W}}_1 - \mathbf{W}_1\mathbf{W}_2)y| \\
&\leq \left(\frac{\varepsilon}{4} + \frac{\varepsilon \cdot (1+\varepsilon/2)}{4}\right) \cdot \left(\|x\|^2_{\mathbf{I}-\mathbf{W}^T\mathbf{W}} + \|y\|^2_{\mathbf{I}-\mathbf{W}\mathbf{W}^T}\right).
\end{aligned}$$

The lemma follows. □

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