Recursive Error Reduction for Regular Branching Programs

Eshan Chattopadhyay
Cornell University, Ithaca, NY, USA

Jyun-Jie Liao
Cornell University, Ithaca, NY, USA

Abstract

In a recent work, Chen, Hoza, Lyu, Tal and Wu (FOCS 2023) showed an improved error reduction framework for the derandomization of regular read-once branching programs (ROBPs). Their result is based on a clever modification to the inverse Laplacian perspective of space-bounded derandomization, which was originally introduced by Ahmadinejad, Kelner, Murtagh, Peebles, Sidford and Vadhan (FOCS 2020).

In this work, we give an alternative error reduction framework for regular ROBPs. Our new framework is based on a binary recursive formula from the work of Chattopadhyay and Liao (CCC 2020), that they used to construct weighted pseudorandom generators (WPRGs) for general ROBPs.

Based on our new error reduction framework, we give alternative proofs to the following results for regular ROBPs of length n and width w, both of which were proved in the work of Chen et al. using their error reduction:

- There is a WPRG with error ε that has seed length

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

- There is a (non-black-box) deterministic algorithm which estimates the expectation of any such program within error $\pm\varepsilon$ with space complexity

$$\tilde{O}(\log(nw) \cdot \log\log(1/\varepsilon)).$$

This was first proved in the work of Ahmadinejad et al., but the proof by Chen et al. is simpler. Because of the binary recursive nature of our new framework, both of our proofs are based on a straightforward induction that is arguably simpler than the Laplacian-based proof in the work of Chen et al.

In fact, because of its simplicity, our proof of the second result directly gives a slightly stronger claim: our algorithm computes a ε-singular value approximation (a notion of approximation introduced in a recent work by Ahmadinejad, Peebles, Pyne, Sidford and Vadhan (FOCS 2023)) of the random walk matrix of the given ROBP in space $\tilde{O}(\log(nw) \cdot \log\log(1/\varepsilon))$. It is not clear how to get this stronger result from the previous proofs.

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization; Theory of computation → Random walks and Markov chains

Keywords and phrases read-once branching program, regular branching program, weighted pseudorandom generator, derandomization

Digital Object Identifier 10.4230/LIPIcs.ITCS.2024.29

Funding Supported by a Sloan Research Fellowship and NSF CAREER award 2045576.

Acknowledgements We want to thank Xin Lyu, Edward Pyne, Salil Vadhan and Hongxun Wu for helpful discussions. We thank anonymous reviewers for helpful comments.

1 Introduction

A central problem in complexity theory is to understand to what extent is randomness useful in space-bounded computation. It is widely conjectured that every randomized algorithm can be made deterministic with only a constant-factor blowup in space, i.e. $\mathbf{BPL} = \mathbf{L}$. A central approach to derandomize $\mathbf{BPL}$ is to construct explicit pseudorandom generators (PRGs) for standard-order read-once branching programs (ROBPs), which we formally define below.

▶ **Definition 1** (ROBPs). *A (standard-order) ROBP B of length n and width w is specified by a start state $v_0 \in [w]$, a set of accept states V_{acc} and n transition functions $B_i : [w] \times \{0,1\} \to [w]$ for i from 1 to n. The ROBP B computes a function $B : \{0,1\}^n \to \{0,1\}$ as follows. Given an input $x \in \{0,1\}^n$, define $v_i = B_i(v_{i-1}, x_i)$, where x_i denotes the i-th bit of x. Then output $B(x) = 1$ if $v_n \in V_{\text{acc}}$, or $B(x) = 0$ otherwise.*

▶ Remark 2. Equivalently, one can view a ROBP B as a directed graph as follows. Consider $n+1$ layers of nodes $L_0, L_1, \ldots, L_n$, each having size w, and label the nodes in each L_i with $[w]$. For every $i \in [n], v \in [w], b \in \{0,1\}$, construct an edge with label b from v in L_{i-1} to $B_i(v,b)$ in L_i. Then the computation of $B(x)$ corresponds to a walk following label x from L_0 to L_n. In this paper we usually consider the equivalent graph view, and we refer to L_i as *layer i*.

▶ **Definition 3** (PRGs). *Let $\mathcal{F}$ be a class of functions $f : \{0,1\}^n \to \{0,1\}$. An ε-PRG for $\mathcal{F}$ is a function $G : \{0,1\}^d \to \{0,1\}^n$ such that for every $f \in \mathcal{F}$,*

$$\left| \mathop{\mathbb{E}}_{x \sim \{0,1\}^n} [f(x)] - \mathop{\mathbb{E}}_{s \sim \{0,1\}^d} [f(G(s))] \right| \leq \varepsilon.$$

We say G ε-fools the class $\mathcal{F}$ if G is an ε-PRG for $\mathcal{F}$. We call d the seed length *of G. We say G is explicit if it can be computed in space $O(d)$.*[1]

It can be shown (via probabilistic method) that there exists a ε-PRG for width-w length-n ROBP with seed length $O(\log(nw/\varepsilon))$, which is optimal. Furthermore, an *explicit* PRG with such seed length would imply $\mathbf{BPL} = \mathbf{L}$. In a seminal work, Nisan [17] constructed an explicit PRG with seed length $O(\log(n) \cdot \log(nw/\varepsilon))$, which is only a $O(\log(n))$ factor away from optimal. Nisan [18] then used this PRG to prove that any problem in $\mathbf{BPL}$ can be deterministically computed in $O(\log^2(n))$ space and $\text{poly}(n)$ time. Another remarkable work by Saks and Zhou [22] also applied Nisan's generator in a non-trivial way to show that any problem in $\mathbf{BPL}$ can be deterministically computed in $O(\log^{3/2}(n))$ space.

1.1 Weighted PRGs

Despite decades of effort, the seed length of Nisan's PRG remains the state-of-the-art for width $w \geq 4$. In fact, even for the $w = 3$ special case, Nisan's seed length remained unbeatable until a recent work by Meka, Reingold and Tal [15] which improved the seed length to $\tilde{O}(\log(n)\log(1/\varepsilon))$. This has motivated researchers to study relaxed notions of PRGs and their applications in the derandomization of $\mathbf{BPL}$. A well-studied notion is that of a hitting set generator (HSG), which is the "one-sided" variant of a PRG.

[1] Throughout this paper, when we say a function f is explicit, it means the function f can be computed in space $O(n)$ where n is the input length.

▶ **Definition 4** (HSGs). *Let $\mathcal{F}$ be a class of functions $f : \{0,1\}^n \to \{0,1\}$. A ε-HSG for $\mathcal{F}$ is a function $G : \{0,1\}^d \to \{0,1\}^n$ such that for every $f \in \mathcal{F}$ s.t. $\mathbb{E}_{x\sim\{0,1\}^n}[f(n)] > \varepsilon$, it holds that $\mathbb{E}_{s\sim\{0,1\}^d}[f(G(s))] > 0$.*

The study of explicit HSGs for ROBPs has a long history, starting from the seminal work by Ajtai, Komlós and Szemerédi [3]. While being weaker than PRGs, explicit constructions of HSGs can still be used to derandomize randomized log-space algorithms with one-sided error (**RL**). In fact, a recent work by Cheng and Hoza [10] shows that an explicit HSG with optimal seed length $O(\log(nw/\varepsilon))$ already implies $\mathbf{BPL} = \mathbf{L}$.

In 2018, Braverman, Cohen and Garg [6] introduced another relaxed notion of PRG called *weighted PRG* (WPRG). In this relaxed notion, each output string of G is further assigned a real weight that can *possibly be negative*.

▶ **Definition 5.** *Let $\mathcal{F}$ be a class of functions $f : \{0,1\}^n \to \{0,1\}$. A ε-WPRG is a pair of functions $(\rho, G) : \{0,1\}^d \to \{0,1\}^n \times \mathbb{R}$ such that for every $f \in \mathcal{F}$,*

$$\left| \mathop{\mathbb{E}}_{x\sim\{0,1\}^n}[f(x)] - \mathop{\mathbb{E}}_{s\sim\{0,1\}^d}[\rho(s)\cdot f(G(s))] \right| \le \varepsilon.$$

Surprisingly, by simply allowing negative weights, [6] showed how to construct an explicit ε-WPRG with seed length

$$\tilde{O}(\log(n)\log(nw) + \log(1/\varepsilon)),$$

which has almost optimal dependence on ε. A sequence of followup work [8, 11, 19, 12] further improved the seed length with simpler WPRG constructions. In particular, Hoza [12] completely removed the hidden log log factors and improve the seed length to $O(\log(n)\log(nw) + \log(1/\varepsilon))$.

It was observed in [6] that ε-WPRGs implies ε-HSGs. In addition, WPRGs seem closer to PRGs than HSGs in the sense that one can use a WPRG to estimate the expectation of a ROBP f by simply enumerating all the seeds. In fact, following a suggestion in [6], [8] proved that a WPRG with good enough bound on the output of ρ can be used in the derandomization framework by Saks and Zhou [22]. Hoza [12] then used the WPRG in [11, 19] to prove that **BPL** can be derandomized in deterministic space $O(\log^{3/2}(n)/\sqrt{\log\log(n)})$. This was the first improvement over Saks and Zhou's decades-old result.

1.2 Regular branching programs

For the original notion of PRGs, while there has been no improvement over Nisan's seed length for general (standard-order) ROBPs, a lot of progress has been made in some restricted families. One important example is the setting of *regular ROBPs*, which is the main focus of this work.

▶ **Definition 6** (Regular ROBPs). *We say a (standard-order) ROBP B is* regular *if for every transition function $B_i : [w] \times \{0,1\} \to [w]$ in B, every state $v \in [w]$ has exactly 2 pre-images.*

An important reason to study this family is that general ROBPs can be reduced to regular ROBPs [20, 5]. In fact, a surprisingly simple proof in a recent work by Lee, Pyne and Vadhan [14] shows that any function that can be computed by a ROBP of length n and width w can also be computed by a regular ROBP of width $O(nw)$.

In 2010, Braverman, Rao, Raz and Yehudayoff [7] proved that the INW generator [13] with proper choices of parameters is in fact a PRG for regular ROBPs with seed length $O(\log(n)\cdot(\log\log(n) + \log(w/\varepsilon)))$. This is better than Nisan's PRG's seed length when

$\log(w/\varepsilon) = o(\log(n))$. More generally, they introduced the "weight" measure for ROBPs and proved that an INW generator with fixed parameters has error proportional to the weight. They then showed that regular ROBPs have smaller weight than general ROBPs when $w \ll n$, which implies their better seed length bound. (See Section 3 for the formal definitions.) Their better PRG construction for "small-weight" ROBPs also turns out to be an important ingredient of the PRG for width-3 ROBPs in [15].

Recently, Ahmadinejad, Kelner, Murtagh, Peebles, Sidford and Vadhan [1] proved a remarkable result that it takes only $\tilde{O}(\log(nw))$ space to estimate the expectation of a regular ROBP B in a non-black-box way. In fact, they designed an algorithm that can estimate the expectation of B to a very high precision without much overhead:

▶ **Theorem 7.** *For every $\varepsilon > 0$ there is a deterministic algorithm which takes a regular ROBP B of length n and width w as input, and computes a value within $\mathbb{E}_x[B(x)] \pm \varepsilon$ in space complexity $\tilde{O}(\log(nw)\log\log(1/\varepsilon))$.*

1.3 Error reduction for regular branching programs

Given the better PRG by [7] in the regular setting, it is natural to ask whether one can get a better WPRG than [12] in the regular setting too. This is in fact plausible because most of the WPRG constructions[2] introduced in Section 1.1 can be viewed as a black-box *error reduction procedures*: given any ε_0-PRG for ROBPs for some "mild error" ε_0 (which we call the "base PRG"), one can construct a ε-WPRG for ROBPs with better dependence on ε. For general standard-order ROBPs, the $O(\log(n)\log(nw) + \log(1/\varepsilon))$ seed length described in Section 3 was obtained by taking Nisan's PRG as the base PRG. Therefore, it is natural to think that one can obtain a better ε-WPRG for regular ROBPs by taking the PRG in [7] as the base PRG instead.

However, it turns out that the intuition is not trivially true, because every known error reduction procedure for general ROBPs requires the "base error" ε_0 to be at most $< 1/n$. When $\varepsilon_0 < 1/n$, the $\tilde{O}(\log(n)\log(w/\varepsilon_0))$ seed length bound in [7] is no better than Nisan's $O(\log(n)\log(nw/\varepsilon_0))$ seed length, so we cannot hope to get any improvement in the seed length of the corresponding WPRG.

This problem was recently solved by Chen, Hoza, Lyu, Tal and Wu [9]. They showed how to exploit the regular property and obtain a reduction from ε-WPRG for regular ROBPs to PRG for regular ROBPs with error $\varepsilon_0 = O(1/\log^2(n))$. As a result, they proved the following theorem.

▶ **Theorem 8** ([9])**.** *There is an explicit ε-WPRG for regular ROBPs with seed length*

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

Following [1, 11, 19, 12], the WPRG construction in [9] is based on the "inverse Laplacian" perspective of small-space derandomization and Richardson iteration. The key step in their construction is to modify the approximated inverse Laplacian based on a structure called "shortcut graph". With the shortcut graph structure, they showed how to apply the potential argument in [7] to get a better bound for ε that is still non-trivial even when $\varepsilon_0 = O(1/\log^2(n))$. Based on the same idea, [9] also showed how to get a simplified proof of the non-black-box derandomization result in [1] (Theorem 7).

[2] This includes [11, 19, 12], and implicitly [8] as we shall see in this paper.

In short, the main purpose of using the shortcut graph idea in [9] is to embed a "binary-recursive-like" structure into the inverse Laplacian analysis. Such a structure makes their analysis compatible with the potential argument in [7]. In order to prove the non-black-box derandomization result in Theorem 7, [9] showed that one can apply a different potential argument based on the notion of "singular-value approximation" (SV approximation) defined in [2].

1.4 Our contribution

While the shortcut graph modification gives a nice structure to the inverse Laplacian analysis, the inverse Laplacian perspective itself is sometimes tricky to work with. In fact, although the proof of Theorem 7 in [9] is simpler than the original proof in [1], they still need to work on a sophisticated matrix seminorm, and the corresponding potential argument requires non-trivial ideas to analyze.

In this work, we give an alternative error reduction framework for regular ROBPs by modifying a WPRG construction by Chattopadhyay and Liao [8]. The advantage of using [8] is that their WPRG construction is *actually binary recursive*, and hence is naturally compatible with the weight argument in [7]. To construct a WPRG for regular branching program that matches the parameter in Theorem 8, we show that the analysis in [8] can be improved in the regular setting based on the weight argument in [7]. Inspired by the proof of Theorem 7 in [9], we also give an alternative proof of Theorem 7 based on the notion of SV approximation. Because of the binary recursive nature of [8], both proofs are relatively straightforward by induction and are arguably simpler than the proofs in [9].

In fact, our proof of Theorem 7 implies a slightly stronger claim (Theorem 32) which might be of independent interest: we can compute an ε-SV approximation of the random walk matrix of any regular ROBP of width w and length n in space $\tilde{O}(\log(nw)\log\log(1/\varepsilon))$. (See [2] for comparison between SV approximation and other notions of approximation.) It is not clear how to obtain this stronger claim from the previous proofs of Theorem 7 [1, 9].

Finally, we show in Appendix D that the Laplacian-based construction in [9] is actually equivalent to the binary recursive construction in [8] that we use in this paper. We note that our proofs of Theorem 7 and Theorem 8 are self-contained and do not rely on this fact.

▶ Remark 9. There are two additional results in [9] which are based on their proof of Theorem 7 and Theorem 8: WPRGs for width-3 ROBPs and WPRGs for unbounded-width permutation ROBPs, both having seed length $\tilde{O}(\log(n)\sqrt{\log(1/\varepsilon)} + \log(1/\varepsilon))$. Our new proofs for Theorem 7 and Theorem 8 can also be plugged into the corresponding parts of their proofs to get the same results.

1.5 Organization

In Section 2 we introduce some general definitions that are used in both the proofs of Theorem 8 and Theorem 7, and give a brief overview of our proofs. In Section 3 we formally prove Theorem 8. In Section 4 we prove Theorem 7.

2 General Setup and Proof Overview

Notation

For $n \in \mathbb{N}$, denote $[n] = \{1, 2, \ldots, n\}$. We write matrices in boldface and use $\mathbf{M}[i, j]$ to denote the entry of matrix $\mathbf{M}$ on the i-th row and the j-th column. We use $\mathbf{I}_w$ to denote the $w \times w$ identity matrix. For a column vector x, we denote the i-th entry of x by $x[i]$.

For every matrix $\mathbf{M} \in \mathbb{R}^{w \times w}$, $\|\mathbf{M}\|_\infty$ denotes the infinity norm $\sup_{\|v\|_\infty=1} \|\mathbf{M}v\|_\infty$ and $\|\mathbf{M}\|$ denotes the 2-norm $\sup_{\|v\|=1} \|\mathbf{M}v\|$. For any alphabet Σ and string $x \in \Sigma^*$, we use $|x|$ to denote the length of x, $x_{[i]}$ to denote the i-th symbol of x and $x_{[\le i]}$ to denote the prefix of x of length x. For any two strings x, y, we use $x \circ y$ to denote the concatenation of x and y.

2.1 ROBPs and matrices

For the rest of this paper, we consider a fixed regular ROBP B of length n and width w specified by transition functions $B_1, \ldots, B_n$. For every $i \in [n]$, and every $b \in \{0,1\}$, define the matrix $\mathbf{M}_i(b) \in \mathbb{R}^{w \times w}$ as

$$\forall u, v \in [w], \mathbf{M}_i(b)[u,v] := \begin{cases} 1 \text{ if } B_i(u,b) = v, \\ 0 \text{ otherwise.} \end{cases}$$

We refer to $\mathbf{M}_i(b)$ as the *transition matrix of B_i on b*. In addition, for every $0 \le \ell < r \le n$ and a string $s \in \{0,1\}^{r-\ell}$, we denote the transition matrix from layer ℓ to layer r on input x as

$$\mathbf{M}_{\ell..r}(s) := \prod_{i=1}^{r-\ell} \mathbf{M}_{\ell+i}(s_i)$$

In this paper we frequently use the following fact:

▶ **Fact 10.** *For every $\ell < m < r$ and $x \in \{0,1\}^{m-\ell}, y \in \{0,1\}^{r-m}$, $\mathbf{M}_{\ell..m}(x)\mathbf{M}_{m..r}(y) = \mathbf{M}_{\ell..r}(x \circ y)$.*

In addition, observe that for a start state $v_0 \in [w]$ and a set of accept state $V_{\text{acc}} \subseteq [w]$, $B(s) = 1$ if and only if there exists $v_n \in V_{\text{acc}}$ s.t. $\mathbf{M}_{0..n}(s)[v_0, v_n] = 1$.

Given the definitions above, we further define $\mathbf{M}_i := \frac{1}{2}(\mathbf{M}_i(0) + \mathbf{M}_i(1))$ which we call the *random walk matrix* of B_i, and define $\mathbf{M}_{\ell..r} := \prod_{i=\ell+1}^{r} \mathbf{M}_i$ which is the random walk matrix from layer ℓ to layer r. Note that $\|\mathbf{M}_{\ell..r}\|_\infty \le 1$ because $\mathbf{M}_{\ell..r}$ is right-stochastic,[3] and we also have $\|\mathbf{M}_{\ell..r}\| \le 1$ because $\mathbf{M}_{\ell..r}$ is doubly-stochastic by the regularity.

Finally, we define v_{st} to be the "start vector" s.t. $v_{\mathsf{st}}[v_0] = 1$ and $v_{\mathsf{st}}[i] = 0$ for every $i \neq v_0$, and v_{ed} to be the "accept vector" s.t. $v_{\mathsf{ed}}[i] = 1$ if $i \in V_{\text{acc}}$ and $v_{\mathsf{ed}}[i] = 0$ otherwise. Then observe that

$$B(s) = v_{\mathsf{st}}^\top \mathbf{M}_{0..n}(s) v_{\mathsf{ed}}$$

and

$$\mathop{\mathbb{E}}_{s \in \{0,1\}^n}[B(s)] = v_{\mathsf{st}}^\top \mathbf{M}_{0..n} v_{\mathsf{ed}}.$$

Given these facts, our goal is to find a "good approximation" of $\mathbf{M}_{0..n}$, denoted by $\widetilde{\mathbf{M}_{0..n}}$, s.t.

$$\left| v_{\mathsf{st}}^\top \mathbf{M}_{0..n} v_{\mathsf{ed}} - v_{\mathsf{st}}^\top \widetilde{\mathbf{M}_{0..n}} v_{\mathsf{ed}} \right| \le \varepsilon.$$

For Theorem 8 we want $\widetilde{\mathbf{M}_{0..n}}$ to correspond to the output of a WPRG with short seed length, while for Theorem 7 we want to make sure that $\widetilde{\mathbf{M}_{0..n}}$ can be implemented in $\widetilde{O}(\log(nw)\log\log(1/\varepsilon))$ space. Because of the different goals, the notions of approximation would also be different in the proofs of Theorem 8 and Theorem 7.

[3] This still holds even when B is not regular.

2.2 Recursion

In this section, we introduce a recursive definition from [8] which we use in both the proofs of Theorem 8 and Theorem 7. Without loss of generality, we assume that n is a power of 2 for the rest of this paper. For ease of notation, we define the set of pairs

$$\mathsf{BS}_n = \{(\ell, r) : \exists i, k \in \mathbb{N} \cup \{0\} \text{ s.t. } \ell = i \cdot 2^k, r = \ell + 2^k \text{ and } 0 \le \ell < r \le n\}.^4$$

Suppose for every $(\ell_0, r_0) \in \mathsf{BS}_n$, we have defined a matrix $\mathbf{M}^{(0)}_{\ell_0..r_0}$ that is a "mild approximation" of $\mathbf{M}_{\ell_0..r_0}$. Then consider the following recursive definition of matrices for every $(\ell, r) \in \mathsf{BS}_n$ and every $k \in \mathbb{N}$, where $m = (\ell + r)/2$:

$$\mathbf{M}^{(k)}_{\ell..r} := \begin{cases} \mathbf{M}_r & \text{if } r - \ell = 1, \\ \sum_{i+j=k} \mathbf{M}^{(i)}_{\ell..m} \cdot \mathbf{M}^{(j)}_{m..r} - \sum_{i+j=k-1} \mathbf{M}^{(i)}_{\ell..m} \cdot \mathbf{M}^{(j)}_{m..r} & \text{otherwise.} \end{cases} \tag{1}$$

The WPRG construction in [8] is exactly a derandomization of the matrix $\mathbf{M}^{(\log(1/\varepsilon))}_{0..n}$, where the base cases $\mathbf{M}^{(0)}_{\ell_0..r_0}$ are generated by Nisan's PRG with error $1/n$. In this paper, we also prove Theorem 8 and Theorem 7 by showing that $\mathbf{M}^{(k)}_{0..n}$ is a good enough approximation of $\mathbf{M}_{0..n}$ (with different choices of the parameter k and base case matrices $\mathbf{M}^{(0)}_{\ell_0..r_0}$).

Now for every $i \ge 0$, define $\mathbf{\Delta}^{(i)}_{\ell..r} := \mathbf{M}^{(i)}_{\ell..r} - \mathbf{M}_{\ell..r}$. The correctness of both [8] and our results relies on the following identity, which was used in the proof of [8, Lemma 15].

▶ **Lemma 11.** *For every $(\ell, r) \in \mathsf{BS}_n$ s.t. $r - \ell > 1$ and $m = (\ell + r)/2$,*

$$\mathbf{\Delta}^{(k)}_{\ell..r} = \sum_{i+j=k} \mathbf{\Delta}^{(i)}_{\ell..m} \cdot \mathbf{\Delta}^{(j)}_{m..r} - \sum_{i+j=k-1} \mathbf{\Delta}^{(i)}_{\ell..m} \cdot \mathbf{\Delta}^{(j)}_{m..r} + \mathbf{\Delta}^{(k)}_{\ell..m} \mathbf{M}_{m..r} + \mathbf{M}_{\ell..m} \mathbf{\Delta}^{(k)}_{m..r}.$$

We briefly sketch how the correctness in [8] was proved based on the lemma above. Suppose the "base PRG" has error ε_0 so that $\left\|\mathbf{\Delta}^{(0)}_{\ell_0..r_0}\right\|_\infty \le \varepsilon_0$. Then one can prove by induction that $\left\|\mathbf{\Delta}^{(k)}_{0..n}\right\|_\infty \le O(n\varepsilon_0)^{k+1}$, i.e. $\mathbf{M}^{(k)}_{0..n}$ is a $O(n\varepsilon_0)^{k+1}$-approximation of $\mathbf{M}_{0..n}$, using the fact that $\|\mathbf{M}_{\ell..r}\|_\infty \le 1$ for every $\ell < r$.

Now observe that the $O(n\varepsilon_0)^{k+1}$ bound is only non-trivial when $\varepsilon_0 < 1/n$. As discussed in the introduction, the seed length of [7] is not better than Nisan's PRG in this parameter regime. Therefore, in the regular setting, even if we can take the base PRG to be the improved PRG in [7], we do not get a better WPRG directly. The main contribution of this work is to give an improved analysis of the error of $\mathbf{M}^{(k)}_{0..n}$ in the regular setting.

2.3 Proof overview

Similar to [9], the reason why we can get an improvement in the regular setting is because a regular ROBP has a bounded "total amount of mixing", no matter how large n is. Our goal is to inductively prove an approximation guarantee that the error of $\mathbf{M}^{(k)}_{\ell..r}$ is *proportional to the amount of mixing* from layer ℓ to layer r. For the proof of WPRG construction (Theorem 8), this statement is formalized based on the "weight" defined in [7]. For the proof of non-black-box derandomization (Theorem 7), this statement is formalized with SV approximation [2]. We defer the formal definitions to later sections, and focus on why this statement gives a better bound.

The first observation is that the last two error terms in Lemma 11 combine nicely. That is, by induction hypothesis we can show that the second last error term $\mathbf{\Delta}^{(k)}_{\ell..m} \mathbf{M}_{m..r}$ is proportional to the amount of mixing from layer ℓ to layer m, and the last error term

$\mathbf{M}_{\ell..m}\mathbf{\Delta}^{(k)}_{m..r}$ is proportional to the amount of mixing from layer m to layer r. Therefore, their sum is proportional to the total amount of mixing from layer ℓ to layer r. Furthermore, we observe that with a proper choice of parameters, the error terms in the first two summations ($\sum_{i+j=k} \mathbf{\Delta}^{(i)}_{\ell..m} \cdot \mathbf{\Delta}^{(j)}_{m..r}$ and $\sum_{i+j=k-1} \mathbf{\Delta}^{(i)}_{\ell..m} \cdot \mathbf{\Delta}^{(j)}_{m..r}$) are actually very small compared to the last two terms, and hence do not affect the total error too much.

Specifically, suppose we already know that the magnitude of $\mathbf{\Delta}^{(i)}_{\ell..m}, \mathbf{\Delta}^{(i)}_{m..r}$ is roughly bounded by $\varepsilon^{(i)}$ for every $i \in \mathbb{N}$, and we want to prove by induction that the magnitude of the new error matrix $\mathbf{\Delta}^{(k)}_{\ell..r}$ is also roughly bounded by $\varepsilon^{(k)}$. We properly choose $\varepsilon^{(i)}$ as in the following lemma, so that the error terms in the first two summations sum up to roughly $\varepsilon^{(k)}/\log(n)$, which is much smaller than the "target error" $\varepsilon^{(k)}$, and hence does not affect the total error too much. We defer the proof of Lemma 12 to Appendix A.[5]

▶ **Lemma 12.** *Let $\gamma < 1/2$, and define $\varepsilon^{(i)} = \frac{\gamma^{i+1}}{10\log(n)(i+1)^2}$. Then for every $k \in \mathbb{N}$ we have*

$$\sum_{i+j=k} \varepsilon^{(i)}\varepsilon^{(j)} + \sum_{i+j=k-1} \varepsilon^{(i)}\varepsilon^{(j)} \le \varepsilon^{(k)}/\log(n).$$

With the choice of parameters above, we can prove that the error of the "level-k approximation" $\mathbf{M}^{(k)}_{\ell..r}$ only grows by a factor of $(1 + 1/\log(n))$ after each recursion. After $\log(n)$ levels of recursion, the error only grows by a constant factor. Therefore, we can choose the "base-case error" $\varepsilon^{(0)}$ to be as small as $O(1/\log(n))$. This allows us to choose base cases with small seed length or space complexity. For the proof of Theorem 8, we choose the base case to be the [7] PRG with error $2^{-\sqrt{\log(1/\varepsilon)}}$. For the proof of Theorem 7 the base cases are generated using derandomized squaring [21, 2].

2.4 Small-space computation

Finally, before we start the formal proofs, we briefly discuss the model of space-bounded computation. We consider the standard model which is a Turing machine with a read-only input tape, a constant number of work tapes, and a write-only output tape. We say an algorithm runs in space s if it uses at most s cells on the *work tapes* throughout the computation. Note that the input length and output length can be larger than s.

Next we recall some basic facts that we will use in space complexity analysis. For parallel composition of algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_t$ we can reuse the work tape and get the following lemma.

▶ **Lemma 13.** *Let $\mathcal{A}_1, \ldots, \mathcal{A}_t$ be algorithms that on input x run in space $s_1, \ldots, s_t$ respectively. Then there exists an algorithm $\mathcal{A}$ that on input x outputs $(\mathcal{A}_1(x), \mathcal{A}_2(x), \ldots, \mathcal{A}_t(x))$ and runs in space $\max_{i\in[t]}(s_i) + O(\log(t))$.*

Furthermore, for sequential composition $\mathcal{A}_1(\mathcal{A}_2(x))$, while we cannot fully store $\mathcal{A}_2(x)$ in the work tape, we can still simulate an input tape containing $\mathcal{A}_2(x)$ by computing the mapping $(x, i) \to \mathcal{A}_2(x)_{[i]}$ instead. (See, e.g., [4, Lemma 4.15].) This implies the following lemma.

▶ **Lemma 14.** *Let $\mathcal{A}_2$ be an algorithm that runs in space s_2 on input x, and $\mathcal{A}_1$ be an algorithm that runs in space s_1 on input $\mathcal{A}_2(x)$. Then there exists an algorithm $\mathcal{A}$ that on input x outputs $\mathcal{A}_1(\mathcal{A}_2(x))$ in space $s_1 + s_2 + O(\log(s_1 + s_2 + |\mathcal{A}_2(x)|))$.*

[5] One can also choose $\varepsilon^{(i)} = \gamma^{i+1}/((2K+1)\log(n))$ where K is an upper bound for k. Then the proof of Lemma 12 becomes straightforward, and it turns out that this does not affect the final results.

We also use the following lemma that can be found in [16, 1].

▶ **Lemma 15.** *Let $\mathbf{M}_1, \ldots, \mathbf{M}_t$ be $w \times w$ real matrices where each entry has bit length at most T. Then $\prod_{i=1}^{t} \mathbf{M}_i$ can be computed in space $O(\log(t)\log(twT))$.*

3 WPRG for regular ROBPs

Using the matrix notation, the weight defined in [7] can be written as follows.[6]

▶ **Definition 16.** *For every vector $y \in \mathbb{R}^w$ and every $i \in [n]$, define the* layer-i weight on y *as*

$$W(i, y) := \sum_{u \in [w]} \sum_{b \in \{0,1\}} \left| (\mathbf{M}_i y)[u] - y[B_i(u, b)] \right|.$$

For every $0 \le \ell < r \le n$, the total weight between layer ℓ and r on y is defined as

$$W(\ell, r, y) := \sum_{i=\ell+1}^{r} W(i, \mathbf{M}_{i..r} y).^7$$

▶ Remark 17. To interpret $W(\ell, r, y)$ with the original description in [7], consider the graph view of ROBPs, and consider y to be the values on the nodes in layer r. Then for every $i \le r$, $\mathbf{M}_{i..r} y$ corresponds to the values on layer i. Observe that each term in the definition of $W(i, \mathbf{M}_{i..r} y)$ corresponds to the "weight" on an edge between layer $i-1$ and i. In consequence, $W(\ell, r, y)$ corresponds to the total weight of the *sub-program between layer ℓ and r* (i.e. the ROBP specified by transition functions $(B_{\ell+1}, \ldots, B_r)$).

The following identity is straightforward by definition:

▶ **Fact 18.** *For every $0 \le \ell < m < r \le n$ and every $y \in \mathbb{R}^w$, $W(\ell, r, y) = W(\ell, m, \mathbf{M}_{m..r} y) + W(m, r, y)$. This also implies $\max(W(\ell, m, \mathbf{M}_{m..r} y), W(m, r, y)) \le W(\ell, r, y)$.*

Given the definition of weight, the main results in [7] imply the following lemmas. Note that Lemma 19 is the only place where regularity is required in this section.[8]

▶ **Lemma 19.** *For every $\ell < r$ and every vector $y \in \mathbb{R}^w$, $W(\ell, r, y) \le w^2 \|y\|_\infty$.*

▶ **Lemma 20.** *For every $\delta > 0$, there exists an explicit PRG $G_0 : \{0,1\}^{d_0} \to \{0,1\}^n$ s.t. for every $0 \le \ell < r \le n$,*

$$\left\| \left(\mathop{\mathbb{E}}_{s \sim \{0,1\}^{d_0}} \left[\mathbf{M}_{\ell..r}(G_0(s)_{[\le r-\ell]}) \right] - \mathbf{M}_{\ell..r} \right) y \right\|_\infty \le \delta W(\ell, r, y).$$

In addition, the seed length is $d_0 = O\left(\log(n)\left(\log\log(n) + \log(w/\delta)\right)\right)$.

Now define $W^* := w^2$, which by Lemma 19 implies $W^* \ge \max_{y:\|y\|_\infty = 1}(W(0, n, y))$. To simplify notation, we define *weight approximation* as follows.

[6] The original results in [7] only consider $y \in [0,1]^w$, but one can easily generalize them to $\mathbb{R}^w$ by shifting and scaling.

[8] To get Lemma 20, we use the fact that $\mathbf{M}_{\ell..r}(\cdot)$ corresponds to the transition matrices of a ROBP of length $(r-\ell)$ and width w, and one can extend it to length n by adding more identity transitions which do not affect the total weight.

▶ **Definition 21.** *For every $0 \le \ell < r \le n$, we say $\widetilde{\mathbf{M}_{\ell..r}}$ is a ε_0-weight approximation of $\mathbf{M}_{\ell..r}$ if*

$$\forall y \in \mathbb{R}^w \quad \left\| \left(\widetilde{\mathbf{M}_{\ell..r}} - \mathbf{M}_{\ell.r} \right) y \right\|_\infty \le \varepsilon_0 \cdot \frac{W(\ell, r, y)}{W^*}.$$

Note that $\widetilde{\mathbf{M}_{\ell..r}}$ being a δ-weight approximation of $\mathbf{M}_{\ell..r}$ also implies $\left\| \widetilde{\mathbf{M}_{\ell..r}} - \mathbf{M}_{\ell..r} \right\|_\infty \le \delta$. Now fix a parameter $\gamma > 0$ to be specified later, and define $\varepsilon^{(i)} = \frac{\gamma^{i+1}}{10(i+1)^2 \log(n)}$ as in Lemma 12. Let G_0 be the PRG in Lemma 20 with parameter $\delta = \varepsilon^{(0)}/(3W^*)$, and for every $(\ell, r) \in \mathsf{BS}_n$ such that $r - \ell > 1$, define

$$\mathbf{M}^{(0)}_{\ell..r} := \mathop{\mathbb{E}}_{s \sim \{0,1\}^{d_0}} \left[\mathbf{M}_{\ell..r}(G_0(s)_{[\le r-\ell]}) \right],$$

which is a $(\varepsilon^{(0)}/3)$-weight approximation by Lemma 20. Then define $\mathbf{M}^{(k)}_{\ell..r}$ recursively as in Equation (1). The following is our main lemma for proving Theorem 8:

▶ **Lemma 22** (main). *For every $k \in \mathbb{N}$, every $y \in \mathbb{R}^w$ and every $(\ell, r) \in \mathsf{BS}_n$, $\mathbf{M}^{(k)}_{\ell..r}$ is a $C_t \varepsilon^{(k)}$-weight approximation of $\mathbf{M}_{\ell..r}$, where $t = \log(r - \ell)$ and $C_t = (1 + 1/\log(n))^t/3$.*

Proof. We prove the lemma by induction over t and k. The first base case $t = 0$ is trivial since $\mathbf{M}^{(k)}_{\ell..r} = \mathbf{M}_{\ell..r}$. The second base case $k = 0$ is also true by definition. For the general case, first we note that the lemma also implies $\left\| \mathbf{\Delta}^{(k)}_{\ell.r} \right\|_\infty \le C_t \varepsilon^{(k)} \le \varepsilon^{(k)}$. Then observe that by Lemma 11 and sub-additivity/sub-multiplicativity of infinity norm, we have

$$\begin{aligned}
\left\| \mathbf{\Delta}^{(k)}_{\ell..r} y \right\|_\infty &\le \sum_{i+j \in \{k-1,k\}} \left\| \mathbf{\Delta}^{(i)}_{\ell..m} \mathbf{\Delta}^{(j)}_{m..r} y \right\|_\infty + \left\| \mathbf{M}_{\ell..m} \mathbf{\Delta}^{(k)}_{m..r} y \right\|_\infty + \left\| \mathbf{\Delta}^{(k)}_{\ell..m} \mathbf{M}_{m..r} y \right\|_\infty \\
&\le \sum_{i+j \in \{k-1,k\}} \left\| \mathbf{\Delta}^{(i)}_{\ell..m} \right\|_\infty \left\| \mathbf{\Delta}^{(j)}_{m..r} y \right\|_\infty + \left\| \mathbf{\Delta}^{(k)}_{m..r} y \right\|_\infty + \left\| \mathbf{\Delta}^{(k)}_{\ell..m} \mathbf{M}_{m..r} y \right\|_\infty \\
&\le \sum_{i+j \in \{k-1,k\}} \left(C^2_{t-1} \varepsilon^{(i)} \varepsilon^{(j)} \cdot \frac{W(m, r, y)}{W^*} \right) \\
&\quad + C_{t-1} \varepsilon^{(k)} \cdot \frac{W(m, r, y) + W(\ell, m, \mathbf{M}_{m..r} y)}{W^*} && \text{(induction)} \\
&\le C_t \varepsilon^{(k)} \cdot \frac{W(\ell, r, y)}{W^*}. \\
& && \text{(by Lemma 12 and Fact 18)}
\end{aligned}$$

In other words, $\mathbf{M}^{(k)}_{\ell..r}$ is a $C_t \varepsilon^{(k)}$-weight approximation of $\mathbf{M}_{\ell..r}$. ◀

Lemma 22 shows that $\mathbf{M}^{(k)}_{0..n}$ is a $\varepsilon^{(k)}$-weight approximation, which also implies that $\left\| \mathbf{M}^{(k)}_{0..n} - \mathbf{M}_{0..n} \right\|_\infty \le \varepsilon^{(k)}$. It remains to construct a WPRG that actually "implements" $\mathbf{M}^{(k)}_{0..n}$. This step is rather standard and is essentially the same as the corresponding step in [9]: to get the seed length as claimed in Theorem 8, we need to further "derandomize" $\mathbf{M}^{(k)}_{0..n}$ using the technique in [11, 19]. In short, we expand the recursive formula for $\mathbf{M}^{(k)}_{0..n}$ and get an "error reduction polynomial" over matrices $\mathbf{M}^{(0)}_{i..j}$. One can show that there are at most $K = n^{O(k)}$ terms in the polynomial, and each term has at most $h = k \log(n)$ factors. Then we can use the INW generator [13] for length h and width w to approximate each term with error $\varepsilon/2K$, which gives us a $\varepsilon/2$-approximation to $\mathbf{M}^{(k)}_{0..n}$. We discuss the details in Section 3.1.

▶ Remark 23. Note that our construction and proof also works for "small-weight ROBPs" in general, if we define $W^* = \max_{B,y:\|y\|_\infty=1}(W(0,n,y))$ instead. This only costs additional $O(\log(n)\log(W^*))$ bits of seed length. We note that this generality is important in some applications, such as the WPRG for width-3 ROBP in [9].

3.1 Final WPRG construction

To simplify notation, we assume without loss of generality that the first output bit of G_0 is unbiased, i.e. $\Pr_{s\in\{0,1\}^{d_0}}\left[G_0(d_0)_{[1]}=1\right]=1/2$.[9] Then we can merge the two different base cases by defining $\mathbf{M}^{(0)}_{r-1..r} := \mathbf{M}_r = \mathbf{M}^{(k)}_{r-1..r} = \mathbb{E}_{s\in\{0,1\}^{d_0}}\left[\mathbf{M}_{r-1..r}(G_0(s)_{[\le 1]})\right]$. Now consider the following notation.

▶ **Definition 24.** *For every $0\le\ell<r\le n$, let $\mathsf{IS}_{\ell..r}$ denote the set of increasing sequences $\mathsf{sq}=(i_0,i_1\ldots,i_h)$ s.t. $\ell=i_0<i_1<\ldots<i_h=r$. We say h is the* length *of* s. *For every $\mathsf{sq}\in\mathsf{IS}_{\ell..r}$, define*

$$\mathbf{M}^{(0)}_{\mathsf{sq}} := \prod_{j=1}^{h}\mathbf{M}^{(0)}_{i_{j-1}..i_j}.$$

Given the notation above, we get the following lemma regarding the expansion of $\mathbf{M}^{(k)}_{0..n}$, which is not hard to prove by induction. For completeness we include a proof in Appendix B.

▶ **Lemma 25.** *For every $k\in\mathbb{N}$ and every $(\ell,r)\in\mathsf{BS}_n$, there is a (multi)set $S\subseteq\mathsf{IS}_{\ell..r}\times\{-1,+1\}$ which satisfies that*

- $\mathbf{M}^{(k)}_{\ell..r}=\sum_{(\mathsf{sq},\sigma)\in S}\sigma\mathbf{M}^{(0)}_{\mathsf{sq}}$.
- $|S|\le(r-\ell)^{2k}$
- *For every $(\mathsf{sq},\sigma)\in S$, the length of sq is at most $k\log(r-\ell)+1$*

In addition, we would need to derandomize each term $\mathbf{M}^{(0)}_{\mathsf{sq}}$ using the following matrix view of INW generator [13], which can be found in, e.g., [6]:

▶ **Lemma 26.** *Let Σ be a finite set of symbols. Suppose for every $i\in[h]$, there is a matrix-valued function $\mathbf{A}_i:\Sigma\to\mathbb{R}^{w\times w}$ which on every input in Σ outputs a stochastic matrix. Then for every $\varepsilon_{\mathrm{INW}}>0$ there exists an explicit function $G_{\mathrm{INW}}:\{0,1\}^d\to\Sigma^h$ such that*

$$\left\|\mathop{\mathbb{E}}_{s\in\{0,1\}^d}\left[\prod_{i=1}^{h}\mathbf{A}_i(G_{\mathrm{INW}}(s)_{[i]})\right]-\prod_{i=1}^{h}\mathop{\mathbb{E}}_{x\in\Sigma}[\mathbf{A}_i(x)]\right\|_\infty\le\varepsilon_{\mathrm{INW}},$$

and $d=O(\log|\Sigma|+\log(h)\log(hw/\varepsilon_{\mathrm{INW}}))$.

Now we are ready to prove Theorem 8.

Proof of Theorem 8. Let $S\subseteq\mathsf{IS}_{0..n}\times[-1,1]$ be the set defined in Lemma 25 s.t. $\mathbf{M}^{(k)}_{0..n}=\sum_{(\mathsf{sq},\sigma)\in S}\sigma\mathbf{M}^{(0)}_{\mathsf{sq}}$. Without loss of generality we can assume that S has size exactly $2^{2k\log(n)}$ by adding dummy sequences with weight $\sigma=0$. In addition, note that there is a enumeration function $E_S:\{0,1\}^{2k\log(n)}\to S$ that can be implemented in space $O(\log(k)\log(n))$ following

[9] To get such a PRG, we can simply take a PRG G'_0 from [7] with $(n-1)$-bit output and define $G_0(b\circ s)=b\circ G'_0(s)$, where b is the first bit.

recursive formula (1).[10] Then consider $G_{\mathrm{INW}} : \{0,1\}^{d_{\mathrm{INW}}} \to \Sigma^h$ in Lemma 26 with $\Sigma = \{0,1\}^{d_0}, h = k\log(n)+1$ and $\varepsilon_{\mathrm{INW}} = \varepsilon/(2|S|)$, then define $d = 2k\log(n) + d_{\mathrm{INW}}$. The final WPRG construction $(\rho, G) : \{0,1\}^d \to \mathbb{R} \times \{0,1\}^n$ is as follows. On any input s,

1. Parse s as $(s_{\mathrm{enum}}, s_{\mathrm{INW}}) \in \{0,1\}^{2k\log(n)} \times \{0,1\}^{d_{\mathrm{INW}}}$
2. Define $((i_0, i_1, \ldots, i_h), \sigma) := E_S(s_{\mathrm{enum}})$.
3. For $j \in [h]$, define $r_j := G_0(G_{\mathrm{INW}}(y)_{[j]}) \in \{0,1\}^n$.
4. Output $(\rho(s), G(s)) := (\sigma|S|, (r_1)_{[\le i_1 - i_0]} \circ (r_2)_{[\le i_2 - i_1]} \circ \ldots \circ (r_h)_{[\le i_h - i_{h-1}]})$.

Next we prove the correctness of G. Observe that

$$\mathop{\mathbb{E}}_{s}[\rho(s)\mathbf{M}_{0..n}(G(s))] = \sum_{((i_0,\ldots,i_h),\sigma)\in S} \sigma \cdot \mathop{\mathbb{E}}_{s_{INW}}\left[\prod_{j=1}^{h} \mathbf{M}_{i_{j-1}..i_j}(G_0(G_{\mathrm{INW}}(s_{\mathrm{INW}})_{[j]}))\right].$$

For every term in the above equation, consider the matrix-valued functions $\mathbf{A}_j : \{0,1\}^{d_0} \to \mathbb{R}^{w\times w}$ s.t. $\mathbf{A}_j(r) = \mathbf{M}_{i_{j-1}..i_j}(G_0(r)_{[\le i_j - i_{j-1}]})$. Note that $\mathbb{E}_r[\mathbf{A}_j(r)] = \mathbf{M}^{(0)}_{i_{j-1}..i_j}$. Then by Lemma 26 we have

$$\left\| \mathop{\mathbb{E}}_{s_{INW}}\left[\prod_{j=1}^{h} \mathbf{M}_{i_{j-1}..i_j}(G_{\mathrm{INW}}(s_{\mathrm{INW}})_j)\right] - \mathbf{M}^{(0)}_{(i_0,i_1,\ldots,i_h)} \right\|_\infty \le \varepsilon/(2|S|),$$

which by the sub-additivity of $\|\cdot\|_\infty$ implies

$$\left\| \mathop{\mathbb{E}}_{s}[\rho(s)\mathbf{M}_{0..n}(G(s))] - \mathbf{M}^{(k)}_{0..n} \right\|_\infty \le \varepsilon/2.$$

We pick suitable γ, k (to be specified later) so that $\varepsilon^{(k)} \le \varepsilon/2$. Then by Lemma 22 we have

$$\left\| \mathop{\mathbb{E}}_{s}[\rho(s)\mathbf{M}_{0..n}(G(s))] - \mathbf{M}^{(k)}_{0..n} \right\|_\infty \le \varepsilon.$$

Then consider the vectors $v_{\mathsf{st}}, v_{\mathsf{ed}} \in \mathbb{R}^w$ corresponding to the start and end states as discussed in Section 2. Observe that $\|v_{\mathsf{st}}\|_1 = 1$ and $\|v_{\mathsf{ed}}\|_\infty \le 1$ by definition. Therefore we have

$$\begin{aligned}
\left| \mathop{\mathbb{E}}_{s\in\{0,1\}^d}[\rho(s)B(G(s))] - \mathop{\mathbb{E}}_{x\in\{0,1\}^n}[B(x)] \right| &= \left| v_{\mathsf{st}}^\top \left(\mathop{\mathbb{E}}_{s}[\rho(s)\mathbf{M}_{0..n}(G(s))] - \mathop{\mathbb{E}}_{x}[\mathbf{M}_{0..n}(x)] \right) v_{\mathsf{ed}} \right| \\
&\le \|v_{\mathsf{st}}\|_1 \left\| \left(\mathop{\mathbb{E}}_{s}[\rho(s)\mathbf{M}_{0..n}(G(s))] - \mathbf{M}_{0..n} \right) v_{\mathsf{ed}} \right\|_\infty \\
&\le \varepsilon.
\end{aligned}$$

Finally we analyze the seed length with an unspecified parameter $0 < \gamma < 1/\log(n)$. Take k to be the minimum integer s.t. $\varepsilon^{(k)} \le \varepsilon/2$. Observe that $\log(1/\varepsilon^{(0)}) = O(1/\gamma)$ and $k = O(\log(1/\varepsilon)/\log(1/\gamma))$. This implies

$$d = d_0 + O(\log(h)\log(hw/\varepsilon) + k\log(n)) = d_0 + \tilde{O}(\log(w/\varepsilon) + k\log(n)).$$

By Lemma 20, we have

$$d_0 = O(\log(n)\log(wW^*/\gamma) + \log(n)\log\log(n)) = \tilde{O}(\log(n)\log(w/\gamma)).$$

[10] That is, we use the first $\lceil \log(2k+1) \rceil$ bits as index to determine a term in the recursive formula, then discard these $\lceil \log(2k+1) \rceil$ bits and recurse. If there's any undefined index, simply return a dummy sequence with weight 0.

Therefore,

$$d = \tilde{O}\left(\log(n)\log(w/\gamma) + \frac{\log(n)\log(1/\varepsilon)}{\log(1/\gamma)} + \log(1/\varepsilon)\right).$$

Taking $\gamma = 2^{-\sqrt{\log(n)}}$, we get

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

Finally, observe that the space complexity is $O(d_0 + \log(k)\log(n) + d_{\mathrm{INW}}) = O(d)$. Therefore the WPRG is explicit. ◀

4 Non-black-box Derandomization for Regular ROBPs

We prove Theorem 7 in this section. Inspired by [9], we use the notion of SV approximation to capture the "amount of mixing". To simplify notation, we define the function $D : \mathbb{R}^{w\times w} \times \mathbb{R}^w \to \mathbb{R}$ to be $D(\mathbf{A}, y) := \|y\|^2 - \|\mathbf{A}y\|^2$, which plays the same role as the weight measure in the proof of Theorem 8. The following fact is straightforward from the definition.

▶ **Fact 27.** $D(\mathbf{B}, y) + D(\mathbf{A}, \mathbf{B}y) = D(\mathbf{AB}, y)$.

The notion of singular value approximation (SV approximation) is defined as follows.

▶ **Definition 28** (SV approximation [2])**.** *Let $\mathbf{W} \in \mathbb{R}^{w\times w}$ be a doubly stochastic matrix. We say $\widetilde{\mathbf{W}}$ is a ε-SV approximation of W if for every $x, y \in \mathbb{R}^w$,*

$$\left|x^\top(\widetilde{\mathbf{W}} - \mathbf{W})y\right| \le \varepsilon \cdot \left(\frac{D(\mathbf{W}^\top, x) + D(\mathbf{W}, y)}{2}\right).$$

Equivalently, for every $x, y \in \mathbb{R}^w$,

$$\left|x^\top(\widetilde{\mathbf{W}} - \mathbf{W})y\right| \le \varepsilon \cdot \left(\sqrt{D(\mathbf{W}^\top, x) \cdot D(\mathbf{W}, y)}\right).$$

The proof of Theorem 7 is very similar to our proof of Theorem 8 in the previous section. First we also need a base case for the different approximation notion. As proved in [2, 9], there is a space-efficient implementation of SV approximation of random walk matrices based on derandomized squaring [21]:

▶ **Lemma 29** ([2, 9])**.** *For every $(\ell, r) \in \mathsf{BS}_n$, there is an algorithm that computes a δ-SV approximation of $\mathbf{M}_{\ell..r}$ in space $\tilde{O}(\log(nw)\log(1/\delta))$. Further, each entry of this approximation matrix has bit length at most $O(\log(n)\log(1/\delta))$.*

We also need the following simple lemma, which can be found in [9]. We include its (short) proof for completeness.

▶ **Lemma 30.** *Suppose $\widetilde{\mathbf{W}}$ is a δ-SV-approximation of $\mathbf{W}$, and let $\mathbf{\Delta} = \widetilde{\mathbf{W}} - \mathbf{W}$. Then*

$$\|\mathbf{\Delta}y\|_2 \le \delta\sqrt{D(\mathbf{W}, y)}.$$

Proof. Observe that

$$\|\mathbf{\Delta}y\|_2^2 = (\mathbf{\Delta}y)^\top \mathbf{\Delta}y \le \delta\sqrt{D(\mathbf{W}, y) \cdot (\|\mathbf{\Delta}y\|_2^2 - \|\mathbf{W}^\top\mathbf{\Delta}y\|_2^2)} \le \delta\sqrt{D(\mathbf{W}, y)} \cdot \|\mathbf{\Delta}y\|.$$ ◀

Now for every $(\ell, r) \in \mathsf{BS}_n$, define $\mathbf{M}^{(0)}_{\ell..r}$ to be a $(\varepsilon^{(0)}/3)$-SV approximation of $\mathbf{M}_{\ell..r}$, and define $\mathbf{M}^{(k)}_{\ell..r}$ using the recursion (Equation (1)). We prove the following lemma which is analogous to Lemma 22:

▶ **Lemma 31** (main). *For every $k \in \mathbb{N}$ and every $(\ell, r) \in \mathsf{BS}_n$, $\mathbf{M}^{(k)}_{\ell..r}$ is a $C_t\varepsilon^{(k)}$-SV approximation of $\mathbf{M}_{\ell..r}$, where $t = \log(r-\ell)$ and $C_t = (1 + 1/\log(n))^t/3$.*

Proof. We again prove the lemma by induction. The base cases $t = 0$ or $k = 0$ are trivial by definition. For the general case, observe that by Lemma 11, for every $x, y \in \mathbb{R}^w$ we have

$$x^\top \mathbf{\Delta}^{(k)}_{\ell..r} y = \sum_{i+j=k} x^\top \mathbf{\Delta}^{(i)}_{\ell..m} \mathbf{\Delta}^{(j)}_{m..r} y - \sum_{i+j=k-1} x^\top \mathbf{\Delta}^{(i)}_{\ell..m} \mathbf{\Delta}^{(j)}_{m..r} y + x^\top \mathbf{M}_{\ell..m} \mathbf{\Delta}^{(k)}_{m..r} y + x^\top \mathbf{\Delta}^{(k)}_{\ell..m} \mathbf{M}_{m..r} y. \tag{2}$$

To bound the first two summations in Equation (2), observe that

$$\begin{aligned}
&\sum_{i+j=k} x^\top \mathbf{\Delta}^{(i)}_{\ell..m} \mathbf{\Delta}^{(j)}_{m..r} y - \sum_{i+j=k-1} x^\top \mathbf{\Delta}^{(i)}_{\ell..m} \mathbf{\Delta}^{(j)}_{m..r} y \\
&\le \sum_{i+j\in\{k-1,k\}} \left\|(\mathbf{\Delta}^{(i)}_{\ell..m})^\top x\right\|_2 \left\|\mathbf{\Delta}^{(j)}_{m..r} y\right\|_2 && \text{(Cauchy-Schwarz)} \\
&\le C_{t-1}^2 \sum_{i+j\in\{k-1,k\}} \varepsilon^{(i)}\varepsilon^{(j)} \sqrt{D\left(\mathbf{M}^\top_{\ell..m}, x\right) D\left(\mathbf{M}_{m..r}, y\right)} && \text{(Lemma 30)} \\
&\le C_{t-1}^2 \cdot \frac{\varepsilon^{(k)}}{\log(n)} \cdot \frac{D\left(\mathbf{M}^\top_{\ell..m}, x\right) + D\left(\mathbf{M}_{m..r}, y\right)}{2} \\
& && \text{(by Lemma 12 and AM-GM)} \\
&\le C_{t-1} \cdot \frac{\varepsilon^{(k)}}{\log(n)} \cdot \frac{D\left(\mathbf{M}^\top_{\ell..r}, x\right) + D\left(\mathbf{M}_{\ell..r}, y\right)}{2} \\
& && (C_{t-1} \le 1 \text{ and } \|\mathbf{M}_{\ell..m}\|_2, \left\|\mathbf{M}^\top_{m..r}\right\|_2 \le 1)
\end{aligned}$$

To bound the last two terms in Equation (2), observe that

$$\begin{aligned}
&x^\top \mathbf{M}_{\ell..m} \mathbf{\Delta}^{(k)}_{m..r} y + x^\top \mathbf{\Delta}^{(k)}_{\ell..m} \mathbf{M}_{m..r} y \\
&\le C_{t-1} \cdot \varepsilon^{(k)} \cdot \frac{D(\mathbf{M}_{m..r}, y) + D(\mathbf{M}^\top_{m..r}, \mathbf{M}^\top_{\ell..m} x) + D(\mathbf{M}_{\ell..m}, \mathbf{M}_{m..r} y) + D(\mathbf{M}^\top_{\ell..m}, x)}{2} \\
&= C_{t-1} \cdot \varepsilon^{(k)} \cdot \frac{D(\mathbf{M}_{\ell..r}, y) + D(\mathbf{M}^\top_{\ell..r}, x)}{2}. && \text{(Fact 18)}
\end{aligned}$$

By summing up the two inequalities, we can conclude that

$$x^\top \mathbf{\Delta}^{(k)}_{\ell..r} y \le C_t \cdot \varepsilon^{(k)} \cdot \frac{D(\mathbf{M}_{\ell..r}, y) + D(\mathbf{M}^\top_{\ell..r}, x)}{2}.$$

Because negating y does not change the bound above, we get

$$\left|x^\top (\mathbf{M}^{(k)}_{\ell..r} - \mathbf{M}_{\ell..r}) y\right| \le C_t \cdot \varepsilon^{(k)} \cdot \frac{D(\mathbf{M}_{\ell..r}, y) + D(\mathbf{M}^\top_{\ell..r}, x)}{2},$$

i.e. $\mathbf{M}^{(k)}_{\ell..r}$ is a $C_t\varepsilon^{(k)}$-SV approximation of $\mathbf{M}_{\ell..r}$. ◀

Finally, let $\gamma = 1/\log(n)$ and $k = O(\log(1/\varepsilon)/\log(1/\gamma))$ be the minimum integer s.t. $\varepsilon^{(k)} \le \varepsilon$. We claim that $\mathbf{M}^{(k)}_{0..n}$ can be implemented in space $\tilde{O}(\log(k)\log(nw))$, which implies the following result:

▶ **Theorem 32.** *There is an algorithm which can compute an ε-SV approximation of the random walk matrix $\mathbf{M}_{0..n}$ in space $\tilde{O}(\log(nw)\log\log(1/\varepsilon))$.*

Note that Theorem 7 is also a direct corollary of this theorem:

Proof of Theorem 7. Compute a $(\varepsilon/\sqrt{w})$-SV approximation of $\mathbf{M}_{0..n}$, denoted by $\widetilde{\mathbf{M}_{0..n}}$. By Theorem 32 this takes space $\tilde{O}(\log(nw)\log\log(1/\varepsilon))$. Then consider $v_{\mathsf{st}}, v_{\mathsf{ed}}$ as defined in Section 2, and output $v_{\mathsf{st}}^\top \widetilde{\mathbf{M}_{0..n}} v_{\mathsf{ed}}$. To prove the correctness, recall that $v_{\mathsf{st}}^\top \mathbf{M}_{0..n} v_{\mathsf{ed}} = \mathbb{E}_x[B(x)]$, which implies

$$\left| v_{\mathsf{st}}^\top \widetilde{\mathbf{M}_{0..n}} v_{\mathsf{ed}} - \mathop{\mathbb{E}}_{x\in\{0,1\}^n}[B(x)] \right| = \left| v_{\mathsf{st}}^\top (\widetilde{\mathbf{M}_{0..n}} - \mathbf{M}_{0..n}) v_{\mathsf{ed}} \right| \le \varepsilon/\sqrt{w}\, \|v_{\mathsf{st}}\|\, \|v_{\mathsf{ed}}\| \le \varepsilon$$

by the fact that $\|v_{\mathsf{st}}\| = 1$ and $\|v_{\mathsf{ed}}\| \le \sqrt{w}$. ◀

▶ Remark 33. Note that $\mathbf{M}^{(k)}_{0..n}$ does not satisfy the original definition of SV approximation in [2] because it is not necessarily doubly stochastic. While every row and column in $\mathbf{M}^{(k)}_{0..n}$ does sum up to 1, some of its entries might be negative.

4.1 Space-efficient implementation

Finally we prove that $\mathbf{M}^{(k)}_{0..n}$ can be implemented in space $\tilde{O}(\log(k)\log(nw))$. Note that a naive implementation of the recursion (Equation (1)) takes at least $O(\log(n)\log(wk))$ space. Furthermore, we cannot naively enumerate each term of $\mathbf{M}^{(k)}_{0..n}$ in its expansion (Lemma 25) either, because there are $n^{O(k)}$ terms in total, which takes at least $O(k\log(n))$ bits to enumerate.

To reduce the space complexity, we will compute $\mathbf{M}_{0..n}$ with a different recursive formula. The intuition of the new recursion is as follows. First observe that each term in the expansion of $\mathbf{M}^{(k)}_{0..n}$ corresponds to a way to put k balls into $2n-1$ bins (indexed by $[2n-1]$), under the constraint that each odd-indexed bin contains at most one ball. To see why this is the case, observe that the original recursion corresponds to the following way to recursively enumerate all the ball-to-bin combinations: first we put $b \in \{0,1\}$ ball in the middle bin (which corresponds to the sign $(-1)^b$), then choose i, j s.t. $i+j = k-b$, and then recursively put i balls in the left $(n-1)$ bins (which corresponds to $\mathbf{M}^{(i)}_{0..n/2}$) and j balls in the right $(n-1)$ bins (which corresponds to $\mathbf{M}^{(j)}_{n/2..n}$).

Then observe that there is a different way to enumerate all the combinations with only $\lceil\log(k)\rceil$ levels of recursion as follows. First decide where the h-th ball is located, where $h = \lceil k/2\rceil$. If it is in an even-indexed bin, also decide how many balls are on the left and how many balls are on the right. Otherwise, there can be only one ball in the selected odd-indexed bin, and the numbers of balls on the left and right are fixed. Then for each choice, recursively enumerate the combinations on the left and right respectively. We claim that there is a corresponding recursive formula for $\mathbf{M}^{(k)}_{0..n}$ which can be implemented in only $\tilde{O}(\log(k)\log(nw))$ space.

To define this recursive formula, first we generalize the definition of $\mathbf{M}^{(k)}_{\ell..r}$ to $(\ell, r) \notin \mathsf{BS}_n$. For any (ℓ, r) s.t. $0 \le \ell < r \le n$, define $\mathsf{LCA}(\ell, r)$ as follows. Let t be the largest integer such that there exists a multiple of 2^t in the range (ℓ, r). Then we define $\mathsf{LCA}(\ell, r)$ to be the unique multiple of 2^t in (ℓ, r).[11] Observe that for $(\ell, r) \in \mathsf{BS}_n$ s.t. $r - \ell > 1$, $\mathsf{LCA}(\ell, r) = (\ell + r)/2$. Therefore, we can generalize the recursion (Equation (1)) to any $r - \ell > 1$ by defining $m = \mathsf{LCA}(\ell, r)$. We also generalize the same recursion to $k = 0, (\ell, r) \notin \mathsf{BS}_n$, so that $\mathbf{M}^{(0)}_{\ell..r} = \mathbf{M}^{(0)}_{\ell..m}\mathbf{M}^{(0)}_{m..r}$.[12] For the degenerate case $\ell = r$ we define $\mathbf{M}^{(k)}_{\ell..r} = \mathbf{I}_w$. Next we want

[11] If there are two consecutive multiples of 2^t in (ℓ, r), then 2^{t+1} divides one of them, which violates the definition of t.

[12] There is an empty summation term $\sum_{i+j=-1}$ which should be treated as 0.

to prove the following identity which naturally gives a recursive algorithm for $\mathbf{M}_{0..n}$. This identity is essentially the recursive enumeration we described above, except that we utilize $\mathbf{M}^{(k)}_{s-1..s} = \mathbf{M}_s$ to get some cancellation which simplifies the recursion.

▶ **Lemma 34.** *For every (ℓ, r) s.t. $r > \ell$ and every $h, k \in \mathbb{N}$ s.t. $h \le k$, we have the following identity:*

$$\mathbf{M}^{(k)}_{\ell..r} = \sum_{s=\ell+1}^{r} \mathbf{M}^{(h-1)}_{\ell..s-1}\mathbf{M}_s\mathbf{M}^{(k-h)}_{s..r} - \sum_{s=\ell+1}^{r-1} \mathbf{M}^{(h-1)}_{\ell..s}\mathbf{M}^{(k-h)}_{s..r}. \tag{3}$$

Surprisingly, this new recursion coincides with the recursion of Richardson iteration from the inverse Laplacian perspective. This shows that the construction in [9] is actually equivalent to the [8] construction that we use in this paper. We briefly discuss this equivalence in Appendix D.

Before we prove the identity, we show that the new recursion does imply an algorithm that runs in space $\tilde{O}(\log(k)\log(nw))$. Consider the algorithm which recursively computes $\mathbf{M}^{(k)}_{\ell..r}$ using the formula in Lemma 34 with $h = \lceil k/2 \rceil$. Observe that the right hand side of Equation (3) involves at most $O(n)$ matrices. In addition, given all the matrices on the right hand side, the computation of $\mathbf{M}^{(k)}_{\ell..r}$ takes only $O(\log(nkwT))$ additional bits, where T is the maximum bit length of all the matrix entries. From Lemma 25 and Lemma 29 we can see that T is at most $\tilde{O}(k\log^2(n))$, so $O(\log(nwkT)) = \tilde{O}(\log(nwk))$. Finally, observe that each matrix on the right hand side has precision parameter at most $\max(h-1, k-h) \le \lfloor k/2 \rfloor$. Therefore the recursion reaches the $\mathbf{M}^{(0)}_{\ell..r}$ base cases after at most $\lceil \log(k) \rceil$ levels. By repeatedly applying Lemma 14 and Lemma 13 we can conclude that the space complexity is at most $\tilde{O}(\log(k)\log(nw)) + O(s_{\text{base}})$, where s_{base} is the maximum space complexity of computing $\mathbf{M}^{(0)}_{\ell..r}$. The base case complexity s_{base} is indeed $\tilde{O}(\log(nw))$, which we prove in Appendix C.

Finally, we prove the identity in Lemma 34.

Proof of Lemma 34. We prove the claim by induction on $r - \ell$. Let $m = \mathsf{LCA}(\ell, r)$. For the base case $r - \ell = 1$, the lemma says $\mathbf{M}^{(k)}_{\ell..r} = \mathbf{M}^{(h-1)}_{\ell..\ell}\mathbf{M}_r\mathbf{M}_{r..r}$, which is trivially true. Next we prove the general case by induction. For each matrix $\mathbf{M}^{(k')}_{\ell'..r'}$ on the right hand side s.t. $\ell' < m < r'$, we expand $\mathbf{M}^{(k')}_{\ell'..r'}$ using (1). Note that $\mathsf{LCA}(\ell', r')$ is also m. In addition, for the $s = m$ term in the first summation we apply the dummy expansion $\mathbf{M}^{(k-h)}_{m..r} = \sum_{a+b=k-h} \mathbf{M}^{(a)}_{m..m}\mathbf{M}^{(b)}_{m..r} - \sum_{a+b=k-h-1} \mathbf{M}^{(a)}_{m..m}\mathbf{M}^{(b)}_{m..r}$. Similarly, for the $s = m+1$ term in the first summation we also expand $\mathbf{M}^{(k-h)}_{\ell..m} = \sum_{a+b=h-1} \mathbf{M}^{(a)}_{\ell..m}\mathbf{M}^{(b)}_{m..m} - \sum_{a+b=h-2} \mathbf{M}^{(a)}_{\ell..m}\mathbf{M}^{(b)}_{m..m}$. After rearranging we get that the right hand side equals to

$$\begin{aligned}
&\sum_{s=\ell+1}^{m} \sum_{a+b=k-h} \mathbf{M}^{(h-1)}_{\ell..s-1}\mathbf{M}_s\mathbf{M}^{(a)}_{s..m}\mathbf{M}^{(b)}_{m..r} && - \sum_{s=\ell+1}^{m-1} \sum_{a+b=k-h} \mathbf{M}^{(h-1)}_{\ell..s}\mathbf{M}^{(a)}_{s..m}\mathbf{M}^{(b)}_{m..r} \\
&- \sum_{s=\ell+1}^{m} \sum_{a+b=k-h-1} \mathbf{M}^{(h-1)}_{\ell..s-1}\mathbf{M}_s\mathbf{M}^{(a)}_{s..m}\mathbf{M}^{(b)}_{m..r} && + \sum_{s=\ell+1}^{m-1} \sum_{a+b=k-h-1} \mathbf{M}^{(h-1)}_{\ell..s}\mathbf{M}^{(a)}_{s..m}\mathbf{M}^{(b)}_{m..r} \\
&+ \sum_{s=m+1}^{r} \sum_{a+b=h-1} \mathbf{M}^{(a)}_{\ell..m}\mathbf{M}^{(b)}_{m..s-1}\mathbf{M}_s\mathbf{M}^{(k-h)}_{s..r} && - \sum_{s=m+1}^{r} \sum_{a+b=h-1} \mathbf{M}^{(a)}_{\ell..m}\mathbf{M}^{(b)}_{m..s}\mathbf{M}^{(k-h)}_{s..r} \\
&- \sum_{s=m+1}^{r} \sum_{a+b=h-2} \mathbf{M}^{(a)}_{\ell..m}\mathbf{M}^{(b)}_{m..s-1}\mathbf{M}_s\mathbf{M}^{(k-h)}_{s..r} && + \sum_{s=m+1}^{r} \sum_{a+b=h-2} \mathbf{M}^{(a)}_{\ell..m}\mathbf{M}^{(b)}_{m..s}\mathbf{M}^{(k-h)}_{s..r} \\
& && - \mathbf{M}^{(h-1)}_{\ell..m}\mathbf{M}^{(k-h)}_{m..r}.
\end{aligned}$$

Note that the first summation in Equation (3) expands to the terms on the left, and the second summation in Equation (3) expands to the terms on the right.

Now we classify all the terms in the first line by b, and take out the right factor $\mathbf{M}^{(b)}_{m..r}$. For any fixed $b \le k-h$ we get the sum

$$\left(\sum_{s=\ell+1}^{m} \mathbf{M}^{(h-1)}_{\ell..s-1}\mathbf{M}_s\mathbf{M}^{(k-b-h)}_{s..m} - \sum_{s=\ell+1}^{m-1} \mathbf{M}^{(h-1)}_{\ell..s}\mathbf{M}^{(k-b-h)}_{s..m}\right)\mathbf{M}^{(b)}_{m..r}.$$

Observe that this is exactly $\mathbf{M}^{(k-b)}_{\ell..m}\mathbf{M}^{(b)}_{m..r}$ by induction. Therefore, we can see that the first line is exactly $\sum_{b=0}^{k-h}\mathbf{M}^{(k-b)}_{\ell..m}\mathbf{M}^{(b)}_{m..r}$. Similarly, we can get that the second line is $-\sum_{b=0}^{k-h-1}\mathbf{M}^{(k-b-1)}_{\ell..m}\mathbf{M}^{(b)}_{m..r}$. For the third and fourth lines, we can also classify the terms by a and take out the left factor $\mathbf{M}^{(a)}_{\ell..m}$ to get to get $\sum_{a=0}^{h-1}\mathbf{M}^{(a)}_{\ell..m}\mathbf{M}^{(k-a)}_{m..r}$ and $-\sum_{a=0}^{h-2}\mathbf{M}^{(a)}_{\ell..m}\mathbf{M}^{(k-a-1)}_{m..r}$ respectively. Finally, collect all the simplified terms (including the only term in the fifth line in the expansion), and we get

$$\begin{aligned}
&\sum_{b=0}^{k-h}\mathbf{M}^{(k-b)}_{\ell..m}\mathbf{M}^{(b)}_{m..r} + \sum_{a=0}^{h-1}\mathbf{M}^{(a)}_{\ell..m}\mathbf{M}^{(k-a)}_{m..r}\\
&- \sum_{b=0}^{k-h-1}\mathbf{M}^{(k-b)}_{\ell..m}\mathbf{M}^{(b)}_{m..r} - \sum_{a=0}^{h-2}\mathbf{M}^{(a)}_{\ell..m}\mathbf{M}^{(k-a-1)}_{m..r} - \mathbf{M}^{(h-1)}_{\ell..m}\mathbf{M}^{(k-h)}_{m..r}\\
&= \sum_{a+b=k}\mathbf{M}^{(a)}_{\ell..m}\mathbf{M}^{(b)}_{m..r} - \sum_{a+b=k-1}\mathbf{M}^{(a)}_{\ell..m}\mathbf{M}^{(b)}_{m..r}\\
&= \mathbf{M}^{(k)}_{\ell..r}.
\end{aligned}$$

◀

References

1. AmirMahdi Ahmadinejad, Jonathan Kelner, Jack Murtagh, John Peebles, Aaron Sidford, and Salil Vadhan. High-precision estimation of random walks in small space. In *FOCS 2020*, 2020.
2. AmirMahdi Ahmadinejad, John Peebles, Edward Pyne, Aaron Sidford, and Salil Vadhan. Singular value approximation and reducing directed to undirected graph sparsification. In *FOCS 2023, to appear*, 2023.
3. Miklós Ajtai, János Komlós, and Endre Szemerédi. Deterministic simulation in LOGSPACE. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 132–140. ACM, 1987. doi:10.1145/28395.28410.
4. Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.
5. Andrej Bogdanov, William M Hoza, Gautam Prakriya, and Edward Pyne. Hitting sets for regular branching programs. In *37th Computational Complexity Conference (CCC 2022)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.
6. Mark Braverman, Gil Cohen, and Sumegha Garg. Pseudorandom pseudo-distributions with near-optimal error for read-once branching programs. *SIAM Journal on Computing*, 49(5):STOC18–242–STOC18–299, 2020. doi:10.1137/18M1197734.
7. Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. Pseudorandom generators for regular branching programs. *SIAM Journal on Computing*, 43(3):973–986, 2014.
8. Eshan Chattopadhyay and Jyun-Jie Liao. Optimal error pseudodistributions for read-once branching programs. In *35th Computational Complexity Conference, CCC 2020*, volume 169 of *LIPIcs*, pages 25:1–25:27. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.
9. Lijie Chen, William Hoza, Xin Lyu, Avishay Tal, and Hongxun Wu. Weighted pseudorandom generators via inverse analysis of random walks and shortcutting. In *FOCS 2023, to appear*, 2023.

10 Kuan Cheng and William M Hoza. Hitting sets give two-sided derandomization of small space. In *35th Computational Complexity Conference (CCC 2020)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

11 Gil Cohen, Dean Doron, Oren Renard, Ori Sberlo, and Amnon Ta-Shma. Error reduction for weighted prgs against read once branching programs. In *36th Computational Complexity Conference (CCC 2021)*, pages 22:1–22:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CCC.2021.22.

12 William M Hoza. Better pseudodistributions and derandomization for space-bounded computation. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

13 Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 356–364, 1994.

14 Chin Ho Lee, Edward Pyne, and Salil Vadhan. On the power of regular and permutation branching programs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023), to appear*, 2023.

15 Raghu Meka, Omer Reingold, and Avishay Tal. Pseudorandom generators for width-3 branching programs. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019*. ACM, 2019. doi:10.1145/3313276.3316319.

16 Jack Murtagh, Omer Reingold, Aaron Sidford, and Salil Vadhan. Derandomization beyond connectivity: Undirected laplacian systems in nearly logarithmic space. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 801–812. IEEE, 2017.

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

18 Noam Nisan. RL ⊆ SC. *Comput. Complex.*, 4:1–11, 1994. doi:10.1007/BF01205052.

19 Edward Pyne and Salil Vadhan. Pseudodistributions that beat all pseudorandom generators. In *36th Computational Complexity Conference (CCC 2021)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

20 Omer Reingold, Luca Trevisan, and Salil Vadhan. Pseudorandom walks on regular digraphs and the RL vs. L problem. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 457–466, 2006.

21 Eyal Rozenman and Salil Vadhan. Derandomized squaring of graphs. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 436–447. Springer, 2005.

22 Michael E. Saks and Shiyu Zhou. $\mathrm{BP}_H\mathrm{Space(S)} \subseteq \mathrm{DSPACE}(S^{3/2})$. *J. Comput. Syst. Sci.*, 58(2):376–403, 1999. doi:10.1006/jcss.1998.1616.

A Proof of Lemma 12

Proof.

$$\begin{aligned}
\sum_{i+j\in\{k-1,k\}} \varepsilon^{(i)}\varepsilon^{(j)} &= \frac{(k+1)^2\varepsilon^{(k)}}{10\log(n)} \cdot \left(\gamma \sum_{i+j\in\{k-1,k\}} \frac{1}{(i+1)^2(j+1)^2}\right) \\
&= \frac{(k+1)^2\varepsilon^{(k)}}{10\log(n)} \cdot \left(\gamma \sum_{i+j\in\{k-1,k\}} \frac{\frac{1}{(i+1)^2}+\frac{1}{(j+1)^2}}{(i+1)^2+(j+1)^2}\right) \\
&\le \frac{(k+1)^2\varepsilon^{(k)}}{10\log(n)} \cdot \left(\gamma \sum_{i+j=k} \frac{\frac{1}{(i+1)^2}+\frac{1}{(j+1)^2}}{(k+2)^2/2} + \sum_{i+j=k-1} \frac{\frac{1}{(i+1)^2}+\frac{1}{(j+1)^2}}{(k+1)^2/2}\right)
\end{aligned}$$

$$\leq \frac{\varepsilon^{(k)}}{10\log(n)} \cdot \left(4(1+\gamma)\sum_{i=0}^{k}\frac{1}{(i+1)^2}\right)$$
$$\leq \frac{\varepsilon^{(k)}}{\log(n)}.$$ ◀

B Proof of Lemma 25

Proof. For $\mathsf{sq}_\ell = (i_0, \ldots, i_{h_\ell}) \in \mathsf{IS}_{\ell..m}$ and $\mathsf{sq}_r = (j_0, \ldots, j_{h_r}) \in \mathsf{IS}_{m..r}$, define

$$\mathsf{sq}_\ell \| \mathsf{sq}_r = (i_0, \ldots, i_{h_\ell} = j_0, \ldots, j_{h_r}).$$

Observe that $\mathsf{sq}_\ell || \mathsf{sq}_r$ has length $h_\ell + h_r$, and that $\mathbf{M}^{(0)}_{\mathsf{sq}_\ell || \mathsf{sq}_r} = \mathbf{M}^{(0)}_{\mathsf{sq}_\ell} \cdot \mathbf{M}^{(0)}_{\mathsf{sq}_r}$.

With the notation above, we prove the lemma by induction over $t := \log(r-\ell)$ and k. The base cases $t = 0$ or $k = 0$ are trivial. Consider the recursive formula (1), and let $S^{(i)}_\ell, S^{(j)}_r$ denote the corresponding sets of $\mathbf{M}^{(i)}_{\ell..m}$ and $\mathbf{M}^{(j)}_{m..r}$ respectively. By distributive law, we get

$$\mathbf{M}^{(k)}_{\ell..r} = \sum_{b\in\{0,1\}} \sum_{i+j=k-b} \sum_{(\mathsf{sq}_\ell,\sigma)\in S^{(i)}_\ell} \sum_{(\mathsf{sq}_r,\sigma)\in S^{(j)}_r} (-1)^b \sigma_\ell \sigma_r \mathbf{M}^{(0)}_{\mathsf{sq}_\ell \| \mathsf{sq}_r}.$$

Therefore, we can define

$$S = \bigcup_{b\in\{0,1\}^{0,1}, i+j=k-b} \{(\mathsf{sq}_\ell \| \mathsf{sq}_r, (-1)^b \sigma_\ell \sigma_r) : (\mathsf{sq}_\ell, \sigma) \in S^{(i)}_\ell \wedge (\mathsf{sq}_r, \sigma) \in S^{(j)}_r\},$$

which satisfies the first condition. Based on the induction hypothesis, the length of each $\mathsf{sq}_\ell \| \mathsf{sq}_r$ is at most $(i+j)(t-1)+2 \leq k(t-1)+2 \leq kt+1$, and the size of S is at most

$$\sum_{i+j\in\{k-1,k\}} \left|S^{(i)}_\ell\right| \left|S^{(j)}_r\right| \leq (2k+1)2^{2k(t-1)} \leq 2^{2kt}.$$ ◀

C Base-case space complexity

In this section we prove the following claim.

▶ **Lemma 35.** *There is an algorithm which for every $\ell < r$ can compute $\mathbf{M}^{(0)}_{\ell..r}$ in space $\tilde{O}(\log(nw))$.*

Proof. We claim that it is possible to output the indices $(\ell_0, r_0) \in \mathsf{BS}_n$ of all the factors $\mathbf{M}^{(0)}_{\ell_0..r_0}$ in the expansion of $\mathbf{M}^{(0)}_{\ell..r}$ in space $O(\log(n))$, and there are only $2\log(n)$ factors. The claim then follows by Lemma 29, Lemma 14 and Lemma 15.

For the base cases $(\ell, r) \in \mathsf{BS}_n$ the claim is straightforward. For $(\ell, r) \notin \mathsf{BS}_n$, first compute $m = \mathsf{LCA}(\ell, r)$, and note that $\mathbf{M}^{(0)}_{\ell..r} = \mathbf{M}^{(0)}_{\ell..m}\mathbf{M}^{(0)}_{m..r}$. For the factors of $\mathbf{M}^{(0)}_{\ell..m}$, while $(\ell, m) \notin \mathsf{BS}_n$, we repeatedly compute $m' = \mathsf{LCA}(\ell, m)$, then output (m', m), and replace m with the value of m'. Eventually when $(\ell, m) \in \mathsf{BS}_n$ we output (ℓ, m).

Note that $\mathbf{M}^{(0)}_{\ell..m} = \mathbf{M}^{(0)}_{\ell..m'}\mathbf{M}^{(0)}_{m'..m}$, so if we can prove that $(m', m) \in \mathsf{BS}_n$, then what we output is exactly the indices of the expansion (except that the output order is reversed, which is easy to deal with). To prove that $(m', m) \in \mathsf{BS}_n$, assume that $m' = c' \cdot 2^{t'}$ and $m = c \cdot 2^t$, for some odd integers c', c. In our algorithm, we always have $m' = \mathsf{LCA}(\ell, m)$ and $m = \mathsf{LCA}(\ell, r')$ for some $r' > m$. Because $\ell < m' < m < r'$, by definition of LCA we must have $t > t'$. We claim that we must have $m = m' + 2^{t'}$, which implies $(m', m) \in \mathsf{BS}_n$.

If this is not the case, then we must have $m' < m' + 2^{t'} < m$ because m is also a multiple of m. Then observe that $m' + 2^{t'}$ is a multiple of $2^{t'+1}$, which contradicts to the fact that $m' = \mathsf{LCA}(\ell, m)$.

Now we have proved how to output the expansion indices of $\mathbf{M}^{(0)}_{\ell..m}$, and for $\mathbf{M}^{(0)}_{m..r}$ the proof is basically the same. Finally, to prove that there are at most $2\log(n)$ factors, observe that every time we replace m with m', the exponent t strictly decreases. Because we always have $0 \le t < \log(n)$, the procedure above can repeat at most $\log(n)$ iterations. Therefore both $\mathbf{M}^{(0)}_{\ell..m}$ and $\mathbf{M}^{(0)}_{m..r}$ have at most $\log(n)$ factors. ◀

D Equivalence between the constructions in [9] and [8]

The construction in [9] is based on the inverse Laplacian perspective of derandomization [1] and preconditioned Richardson iteration, which we briefly recap as follows. Consider the block matrix $\mathbf{W} = (\mathbf{R}^{w\times w})^{(n+1)\times(n+1)}$ s.t. $\mathbf{W}[i-1, i] = \mathbf{M}_i$ for every $i \in [n]$, and other entries of $\mathbf{W}$ are 0. The Laplacian is defined as $\mathbf{L} := \mathbf{I} - \mathbf{W}$, which is an invertible matrix. It can be shown that each entry $\mathbf{L}^{-1}[i, j]$ in the inverse Laplacian $\mathbf{L}^{-1}$ is exactly $\mathbf{M}_{i..j}$ (where $\mathbf{M}_{i..i} = \mathbf{I}_w$ and $\mathbf{M}_{i..j} = 0$ if $i > j$).

To approximate $\mathbf{M}_{0..n}$ within error ε, [9] followed the preconditioned Richardson iteration approach, which first constructs a "mild approximation" of $\mathbf{L}^{-1}$ denoted by $\widetilde{\mathbf{L}^{-1}}$. Their construction of $\widetilde{\mathbf{L}^{-1}}$ is based on the shortcut graph structure, and one can verify that their $\widetilde{\mathbf{L}^{-1}}[i, j]$ is actually the same as $\mathbf{M}^{(0)}_{i..j}$ defined in this paper.

Given $\mathbf{L}^{-1}$, the construction based on Richardson iteration is defined as $(\mathbf{L}^{-1})^{(k)} = \widetilde{\mathbf{L}^{-1}} \sum_{i=0}^{k} (\mathbf{I} - \mathbf{L}\widetilde{\mathbf{L}^{-1}})^i$, and $(\mathbf{L}^{-1})^{(k)}[0, n]$ is the final output. Observe that this formula satisfies the recursion $(\mathbf{L}^{-1})^{(k)} = \widetilde{\mathbf{L}^{-1}} + (\mathbf{L}^{-1})^{(k-1)}(\mathbf{I} - \mathbf{L}\widetilde{\mathbf{L}^{-1}})$. In addition, observe that for every $i < j$, $(\mathbf{I} - \mathbf{L}\widetilde{\mathbf{L}^{-1}})[i, j] = \mathbf{M}_{i+1}\mathbf{M}^{(0)}_{i+1..j} - \mathbf{M}^{(0)}_{i..j}$,[13] and other entries of $\mathbf{I} - \mathbf{L}\widetilde{\mathbf{L}^{-1}}$ are 0. Therefore, for every $\ell \le r$ we have

$$(\mathbf{L}^{-1})^{(k)}[\ell, r] = \mathbf{M}^{(0)}_{\ell..r} + \sum_{s=\ell+1}^{r} (\mathbf{L}^{-1})^{(k-1)}[\ell, s-1] \left(\mathbf{M}_s\mathbf{M}^{(0)}_{s..r} - \mathbf{M}^{(0)}_{s-1..r}\right).$$

Comparing it with the special case $h = k$ of Lemma 34:

$$\mathbf{M}^{(k)}_{\ell..r} = \sum_{s=\ell+1}^{r} \mathbf{M}^{(k-1)}_{\ell..s-1}\mathbf{M}_s\mathbf{M}^{(0)}_{s..r} - \sum_{s=\ell+1}^{r-1} \mathbf{M}^{(k-1)}_{\ell..s}\mathbf{M}^{(0)}_{s..r}.$$

Using the fact that $(\mathbf{L}^{-1})^{(k-1)}[\ell, \ell] = \mathbf{I}$, it is relatively easy to prove via induction that $(\mathbf{L}^{-1})^{(k)}[\ell, r] = \mathbf{M}^{(k)}_{\ell,r}$.

[13] This is in fact the *local consistency error* defined in [10], as observed in [12].