Electronic Colloquium on Computational Complexity, Report No. 27 (2025)

Weighted Pseudorandom Generators for Read-Once Branching Programs via Weighted Pseudorandom Reductions

Kuan Cheng * Ruiyang Wu †

Abstract

We study weighted pseudorandom generators (WPRGs) and derandomizations for read-once branching programs (ROBPs), which are key problems towards answering the fundamental open question $\mathbf{BPL} \stackrel{?}{=} \mathbf{L}$. Denote n and w as the length and the width of a ROBP. We have the following results.

For standard ROBPs, there exists an explicit ε-WPRG with seed length

$$O\left(\frac{\log n \log(nw)}{\max\{1, \log\log w - \log\log n\}} + \log w \left(\log\log\log w - \log\log\max\left\{2, \frac{\log w}{\log n/\varepsilon}\right\}\right) + \log(1/\varepsilon)\right).$$

When $n = w^{o(1)}$, this is better than the constructions in [Hoz21, CDR+21, PV21, CL20].

For permutation ROBPs with unbounded widths and single accept nodes, there exists an explicit ε-WPRG with seed length

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

This slightly improves the result of [CHL+23].

For regular ROBPs with $n \leq 2^{O(\sqrt{\log w})}, \varepsilon = 1/\operatorname{poly} w$, we give a derandomization within space $O(\log w)$, i.e. in $\mathbf{L}$ exactly. This is better than previous results of [AKM+20, CHL+23, CL24] in this regime.

Our main method is based on a recursive application of weighted pseudorandom reductions, which is a natural notion that is used to simplify ROBPs.

1 Introduction

Randomness is a fundamental resource in computation, but is it essential? A key conjecture in space-bounded computation is that randomized algorithms in the complexity class **BPL** can be efficiently simulated by deterministic logspace algorithms, i.e. $\mathbf{BPL} = \mathbf{L}$. A central approach toward addressing this conjecture is the derandomization of standard-order read-once branching programs (ROBPs), which is usually defined as the following.

Definition 1.1 (Read-once branching programs (ROBP))**.** *A read-once branching program f of length n, width w and alphabet size $|\Sigma| = 2^s$ is a directed acyclic graph with $n+1$ layers $V_0, \ldots, V_n$. For any layer V_i except V_n, each node $v \in V_i$ has 2^s outgoing edges to nodes in V_{i+1}. These edges are labeled by distinct symbols in Σ. There exists a unique start node $v_{start} \in V_0$ and a set of accept nodes $V_{accept} \subset V_n$. Given an input $x \in \Sigma^n$, the computation of $f(x)$ is defined as: $f(x) = 1$, if there exists a unique path $v_{start}, v_1, \ldots, v_n$ such that the edge between v_i and v_{i+1} is labeled by x_i, and $v_n \in V_{accept}$; $f(x) = 0$ otherwise.*

*CFCS, School of Computer Science, Peking University. ckkcdh@pku.edu.cn.
†CFCS, School of Computer Science, Peking University. wuruiyang@stu.pku.edu.cn.

ISSN 1433-8092

Every problem in **BPL** can be reduced to approximating $\mathbb{E}f$ for a corresponding ROBP f. A classical method for derandomizing ROBPs is constructing pseudorandom generators (PRGs).

Definition 1.2 (PRG)**.** *Let $\mathcal{F}$ be a class of ROBPs $f : (\{0,1\}^s)^n \to \{0,1\}$. An ε-PRG for $\mathcal{F}$ is a function $G : \{0,1\}^d \to (\{0,1\}^s)^n$ such that for every $f \in \mathcal{F}$, we have*

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

*The input length d is called the seed length of the PRG. We say that G is **explicit** if it can be computed in space $O(d)$ and time $\mathrm{poly}(d, n)$.*

Using the probabilistic method, one can show the existence of a non-uniform ε-PRG for standard-order ROBPs of length n, width w, and alphabet size 2^s, with an optimal seed length of $O(s + \log(nw/\varepsilon))$. However, constructing explicit PRGs that have short seed lengths turns out to be an exceptional challenge. In a seminal work, Nisan [Nis92a] constructed an explicit ε-PRG for ROBPs of length n, width w, and binary alphabet $\{0,1\}$, with a seed length of $O(\log n \log(nw/\varepsilon))$. Building on this result, Saks and Zhou [SZ99] developed a celebrating algorithm to derandomize **BPL**, within $O(\log^{3/2} n)$ space deterministically. Nisan [Nis92b] also used [Nis92a] to show that **BPL** $\subseteq$ **SC**. Impagliazzo, Nisan, and Wigderson [INW94] generalized the construction of [Nis92a] by using expanders, to fool more general models in network communication. In the meantime, for short ROBPs, Nisan and Zuckerman [NZ96] gave another remarkable PRG that has a better seed length for wide but short ROBPs, i.e. with seed length $O(\log w)$ when $n = \mathrm{poly}\log w, \varepsilon = 2^{-\log^{0.99} w}$. Armoni [Arm98] further extended [Nis92a, NZ96] to construct an improved PRG with seed length[1] $O\left(\frac{\log n \log(nw/\varepsilon)}{\max\{1, \log\log w - \log\log(n/\varepsilon)\}}\right)$.

1.1 Weighted Pseudorandom Generators

Despite years of research, the challenging problem of constructing better PRGs for general ROBPs remains open. Nisan's generator is still the best explicit PRG for ROBPs when lengths and widths are large, e.g. $n = w = 1/\varepsilon$. However, PRGs are not the only black-box solution to derandomization. Weighted pseudorandom generator (WPRG) and Hitting set generator (HSG) can also do derandomizations.

Braverman, Cohen, and Garg [BCG19] introduced WPRG which is a weaker notion than PRG.

Definition 1.3 (WPRG)**.** *Let $\mathcal{F}$ be a class of ROBPs $f : (\{0,1\}^s)^n \to \{0,1\}$. A W-bounded ε-WPRG for $\mathcal{F}$ is a function $(G, w) : \{0,1\}^d \to (\{0,1\}^s)^n \times \mathbb{R}$ such that for every $f \in \mathcal{F}$, we have*

$$\left|\mathbb{E}_{x\in(\{0,1\}^s)^n} f(x) - \sum_{r\in\{0,1\}^d} \left[\frac{1}{2^d} w(r) \cdot f(G(r))\right]\right| \leq \varepsilon,$$
$$\forall r, |w(r)| \leq W.$$

*The input length d is called the seed length of the WPRG. We say that (G, w) is **explicit** if it can be computed in space $O(d)$.*

As shown in [BCG19], under this notion, the seed length can be better than those of previous PRGs in the sense that for ε there is only an isolated addend in the seed length. Chattopadhyay

[1] [Arm98] needs to use an extractor with seed length optimal up to constant factors which is discovered later than [Arm98], e.g. [GUV09].

and Liao [CL20] further improved this isolated addend to be optimal $O(\log(1/\varepsilon))$. Meanwhile, Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Salil Vadhan [AKM+20] developed another method based on Richardson Iterations to reduce errors in derandomizing random walks for certain specific graphs. Then Cohen, Doron, Renard, Renard, Sberlo, and Ta-Shma [CDR+21], and also Pyne and Vadhan [PV21] further developed this technique to be a black-box error reduction from PRGs to WPRGs. Hoza [Hoz21] improved this construction to attain seed length $O(\log n \log(nw) + \log(1/\varepsilon))$ for the binary alphabet.

Inspired by error reductions used in these WPRG constructions, Hoza [Hoz21] improved the deranomization of Saks and Zhou [SZ99] to be $\mathbf{BPL} \subseteq \mathbf{DSPACE}\left(\frac{\log^{3/2} n}{\sqrt{\log\log n}}\right)$. Cohen, Doron, Sberlo, and Ta-Shma [CDST23], also Pyne and Putterman [PP23], showed that ROBPs with medium width $w = 2^{O(\sqrt{\log n})}$ can be drandomized in $\tilde{O}(\log n)$ space. Cheng and Wang [CW24] showed that $\mathbf{BPL} \subseteq$ logspace-uniform $\mathbf{AC}^1$.

HSG is an even weaker notion than WPRG. However, it is also a powerful tool in derandomization.

Definition 1.4 (HSG)**.** *Let $\mathcal{F}$ be a class of ROBPs $f : (\{0,1\}^s)^n \to \{0,1\}$. An ε-HSG for $\mathcal{F}$ is a function $H : \{0,1\}^d \to (\{0,1\}^s)^n$ such that for every $f \in \mathcal{F}$, if $\mathbb{E}_{x\in(\{0,1\}^s)^n} f(x) \geq \varepsilon$, then $\exists r \in \{0,1\}^d, f(H(r)) = 1$.*

*The input length d is called the seed length of the HSG. We say that H is **explicit** if it can be computed in space $O(d)$.*

One can use HSG to find an accepting path when the acceptance probability is significant. Actually, HSG is more powerful than this. Cheng and Hoza [CH22] showed how to approximate the acceptance probabilities of ROBPs of length n, width w using an HSG for ROBPs with significantly larger width $O(\text{poly}(nw/\varepsilon))$ and length $O(\text{poly}(nw/\varepsilon))$. Pyne, Raz, and Zhan [PRZ23] further extended the method to give a deterministic sampler for such a task. Constructing better HSGs is a also challenging task. Hoza and Zuckerman gave the current best HSG with seed length $O\left(\frac{\log n \log(nw)}{\max\{1,\log\log w - \log\log n\}} + \log(1/\varepsilon)\right)$ for binary alphabet.

1.2 WPRG for short-wide standard ROBPs

Although significant progress has been made in the study of WPRGs for general ROBPs with equal length and width, the known WPRGs for the regime of short-wide ROBPs, where $n = \text{poly}(\log w)$, has no advantage compared to early works [Nis92a, INW94, NZ96, Arm98]. For binary alphabet, NZ PRG [NZ96] and Armoni's PRG [Arm98] have optimal seed lengths of $O(\log w)$ when $\varepsilon = 2^{-\log^{1-v} w}$, where v is any constant in $(0, 1)$. When we need a smaller error such as $\varepsilon = 1/\text{poly}(w)$, the WPRGs by [CDR+21, Hoz21, PV21] have seed lengths deteriorates to $O(\log w \log\log w)$ which is the same as that of [Nis92a, INW94, Arm98]. This naturally raises the question: can we construct a WPRG for short-wide ROBPs with seed length $O(\log w + \log(1/\varepsilon))$ for this regime? The question is also interesting for larger n, i.e. can we have a WPRG construction with seed length meeting that of the HSG construction in [HZ20].

In this paper, we make progress for answering these questions by constructing a ε-WPRG for ROBPs of width w and length n with better seed length. Specifically, we achieve the following results.

Theorem 1.5. *For every $n, w \in \mathbb{N}$, $\varepsilon \in (0, 1)$, with $n \geq \log w$, there exists an explicit ε-WPRG with seed length*

$$O\left(\frac{\log n \log(nw)}{\max\{1, \log\log w - \log\log n\}} + \log w\left(\log\log\log w - \log\log\max\left\{2, \frac{\log w}{\log n/\varepsilon}\right\}\right) + \log(1/\varepsilon)\right)$$

for the class of ROBPs of width w, length n and alphabet $\{0,1\}$.

This WPRG has a better seed length than Armoni's PRG [Arm98, KNW08]. Also, this slightly outperforms [Hoz21, CDR+21, PV21, CL20], when $n << \mathrm{poly}(w)$.

For the special case $n = \mathrm{poly}\log w$, we have the following result.

Theorem 1.6. *Given any constant $C > 0$, for every $w, s \in \mathbb{N}$ and $\varepsilon \in (0,1)$, there exists an explicit ε-WPRG with seed length*

$$O(s + \log w \log\log\log w + \log(1/\varepsilon))$$

for the class of ROBPs of width w, length $n = \log^C w$ and alphabet $\{0,1\}^s$.

Table 2 and Table 1 summarize the seed length of ε-PRGs, ε-HSGs and ε-WPRGs for short-wide general ROBPs.

Seed length	Type	Reference
$O\left(\frac{\log n \log(nw/\varepsilon)}{\log\log w - \log\log(n/\varepsilon)}\right)$	PRG	[Arm98, KNW08]
$O(\log n \log(nw) + \log(1/\varepsilon))$	WPRG	[Hoz21]
$O\left(\frac{\log n \log(nw)}{\log\log w - \log\log n} + \log(w/\varepsilon)\log\log_n(1/\varepsilon)\right)$	WPRG	[CDR+21]
$O\left(\frac{\log n \log(nw)}{\log\log w - \log\log n} + \log(1/\varepsilon)\right)$	HSG	[HZ20]
$O\left(\frac{\log n \log(nw)}{\log\log w - \log\log n} + \log w \log\log\log w + \log(1/\varepsilon)\right)$	WPRG	This work

Table 1: Comparison of seed length of ε-PRGs, ε-HSGs and ε-WPRGs for general ROBPs of width w, length $n \le \sqrt{w}$, and alphabet $\{0,1\}$, where $\varepsilon \le 1/\mathrm{poly}(w)$.

Seed length	Type	Reference	Note
$O(\log w)$	PRG	[NZ96]	$\varepsilon = 2^{-\log^{1-v} w}$
$O(\log(w/\varepsilon)\log\log w)$	PRG	[NZ96],[INW94],[Nis92a]	
$O(\log w \log\log w + \log(1/\varepsilon))$	WPRG	[Hoz21],[CDR+21]	
$O(\log w + \log(1/\varepsilon))$	HSG	[HZ20]	
$O(\log w \log\log\log w + \log(1/\varepsilon))$	WPRG	This work	
$O(\log w + \log(1/\varepsilon))$	PRG	folklore	Optimal; non-explicit

Table 2: Comparison of seed length of ε-PRGs, ε-HSGs and ε-WPRGs for short-wide ROBPs of width w, length $n = \mathrm{poly}(\log w)$, and alphabet $\{0,1\}$.

1.3 WPRG for unbounded-width permutation ROBPs with a single accepting node

Recently researchers have also spent much effort on derandomization for special classes of ROBPs. One such class is permutation ROBPs, where the transition functions between layers are permutations.

Definition 1.7 (permutation ROBP)**.** *A (standard-order) permutation ROBP is a standard-order ROBP f, where for every $i \in [n]$ and $x \in \{0,1\}^s$, the transition matrix from V_i to V_{i+1} through edges labeled by x, is a permutation matrix in $\mathbb{R}^{w\times w}$.*

Early work on PRGs for permutation ROBPs [BV10, De11, KNP11, Ste12, RSV13, CHHL19] focus on the constant width case. Inspired by the progress on derandomizing squares [RV05, AKM+20, APP+23], recent studies focus on unbounded-width permutation ROBPs with a single accepting node [HPV21a, PV21, BHPP22, CHL+23]. Hoza, Pyne and Vadhan [HPV21a] designed an ε-PRG for unbounded-width permutation ROBPs with seed length $\widetilde{O}(\log n \log(1/\varepsilon))$ for binary alphabet. They also proved that any PRG for this class must have seed length $\widetilde{\Omega}(\log n \log(1/\varepsilon))$. For the WPRG case, Pyne and Vadhan [PV21] designed a ε-WPRG with seed length $\widetilde{O}(\log n\sqrt{\log(n/\varepsilon)}+\log(1/\varepsilon))$. This was improved by Chen, Hoza, Lyu, Tal, and Wu [CHL+23] to[2] $\widetilde{O}(\log n\sqrt{\log(1/\varepsilon)}+\log(1/\varepsilon))$.

Our idea in WPRG construction for general ROBPs can also be used to slightly improve the WPRG construction for permutation ROBPs. Specifically, we have the following result.

Theorem 1.8. *For every $n, s \in \mathbb{N}$ and $\varepsilon \in (0,1)$, there exists an explicit ε-WPRG with seed length*

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

for the class of permutation ROBPs of length n and alphabet $\{0,1\}^s$ with a single accepting node.

When the error is small enough, i.e. $\varepsilon \le 2^{-\log^2 n}$, our WPRG has a seed length optimal up to a constant factor. The comparison of our result with the previous results is shown in Table 3.

Seed length	Type	Reference	Note
$\widetilde{O}(\log n \log(1/\varepsilon))$	PRG	[HPV21a]	
$\widetilde{O}(\log n\sqrt{\log(n/\varepsilon)}+\log(1/\varepsilon))$	WPRG	[PV21]	
$\widetilde{O}(\log n\sqrt{\log(1/\varepsilon)}+\log(1/\varepsilon))$	WPRG	[CHL+23]	
$O\left(\log n\left(\log\log n+\sqrt{\log(1/\varepsilon)}\right)+\log(1/\varepsilon)\right)$	WPRG	This work	
$\widetilde{O}(\log n \log(1/\varepsilon))$	PRG	[HPV21a]	lower bound

Table 3: Comparison of seed length of ε-PRGs and ε-WPRGs for unbounded-width permutation ROBPs of length n and alphabet $\{0,1\}$ with one accepting node.

1.4 Derandomization for short regular ROBPs

Regular ROBPs is another interesting sub-class of standard ROBPs.

Definition 1.9 (regular ROBP)**.** *A regular ROBP is a standard-order ROBP f, where for every $i \in [n]$, the bipartite graph induced by the nodes in layers V_{i-1} and V_i is a regular graph.*

Apparently regular ROBPs are special ROBPs, and permutation ROBPs are special regular ROBPs. There is an extensive body of work focused on constructing HSGs, WPRGs, and PRGs for regular ROBPs [BRRY14, BHPP22, De11, RSV13, CL24, CHL+23]. Additionally, Lee, Pyne, and Vadhan [LPV23] demonstrated that any standard ROBPs of length n, width w and alphabet $\{0,1\}$ can be equivalently computed by a regular ROBP of width $O(nw)$. These results also have shed light on derandomizing standard ROBPs and space-bounded computations.

PRGs and WPRGs constructed for regular ROBPs naturally induce derandomization algorithms for this class. However, dedicated derandomization algorithms achieve better performance. Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan [AKM+20] derandomized regular

[2]The detailed parameter of [CHL+23] is $O\left(\log n\sqrt{\log(1/\varepsilon)}\sqrt{\log\log(n/\varepsilon)}+\log(1/\varepsilon)\log\log(n/\varepsilon)\right)$.

ROBPs of length n, width w and alphabet $\{0,1\}$ within error ε, using space $\widetilde{O}(\log(nw)\log\log(1/\varepsilon))$. Subsequently, Chen, Hoza, Lyu, Tal, and Wu [CHL+23] achieved the same result with a simplified algorithm. Chattopadhyay and Liao [CL24] constructed another alternate algorithm with the same space complexity.

In this paper, we focus on derandomization algorithms for short-wide regular ROBPs and construct a derandomization algorithm with improved space complexity for this regime, by adapting our WPRG for permutation ROBPs. In fact, our result can be more generally stated as the following.

Theorem 1.10. *There exists an algorithm that takes a regular ROBP f of length n, width w and alphabet $\{0,1\}^s$ and a parameter $\varepsilon > 0$ as input, and outputs a value that approximates $\mathbb{E}[f]$ within error ε. The algorithm runs in $O(s + \log n(\log\log n + \sqrt{\log(w/\varepsilon)}) + \log(w/\varepsilon))$ bits of space.*

Notice that when $n \le 2^{O(\log^{1/2} w)}$ and $\varepsilon = 1/\operatorname{poly} w$ our algorithm achieves a space complexity of $O(s + \log w)$, which is optimal up to a constant factor, i.e. in $\mathbf{L}$ exactly. The comparison of our result with the previous results is shown in Table 4.

Space complexity when $n \le 2^{\log^{1/2} w}$ and $\varepsilon = 1/\operatorname{poly}(w)$	Reference
$\widetilde{O}(\log w \log\log w)$	[AKM+20, CHL+23, CL24]
$O(\log w)$	This work

Table 4: Comparison of space complexity of derandomization algorithms for short-wide regular ROBPs of length n, width w and alphabet $\{0,1\}$.

1.5 Technical Overview

Our general method is inspired first by an observation about the error reduction technique introduced in [CDR+21, PV21]. Recall that in [CDR+21, PV21], a given ROBP $\{A_i\}_{i=1}^n \in \mathbb{R}^{w\times w}$, the product $A = \Pi_{i=1}^n A_i$ can be approximated by a weighted summation of a sequence of iterated matrix multiplications,

$$\left\| A - \sum_{i\in[K]} \sigma_i \cdot B_{0,n_{i,1}} B_{n_{i,1},n_{i,2}} \dots B_{n_{i,k-1},n_{i,k}} \right\| \le \varepsilon_0^k \cdot (n+1).$$

where $\sigma_i \in \{-1,0,1\}$, $K = n^{O(k)}$, $\{B_{i,j}\}_{i,j=0}^n$ can be any family of matrices such that $\|B_{i,j} - A_{i+1}\dots A_j\| \le \varepsilon_0/(n+1)$, and $\|\cdot\|$ can be any sub-multiplicative norm such that $\|A_i\| \le 1$. Then one can deploy some inner PRGs, which are some well-known PRGs, e.g. INW generators, for each $B_{i,j}$ such that when the ROBP reads the outputs of these inner PRGs, the corresponding approximation matrix is a valid $B_{i,j}$. Further one can use an outer PRG with a large alphabet, to generate seeds for inner PRGs. This will give the WPRG of [CDR+21, PV21].

An observation is that $B_{0,n_{i,1}} B_{n_{i,1},n_{i,2}} \dots B_{n_{i,k-1},n_{i,k}}$ can be viewed as a shorter ROBP with a large alphabet, while the expectation of the original branching program can be approximated as the sum of weighted expectations of shorter branching programs. So an initial thought is that if one can somehow recursively apply this procedure, to further reduce the length of the ROBP, until the length becomes a constant, then by using true randomness for constant length ROBPs, one can attain a WPRG. Of course, there are problems with this initial thought: first, the alphabet size may increase quickly; second, one needs to make sure the length can indeed be shorter each time we apply the reduction; third, the overall weight needs to be bounded. We will exhibit how to resolve these problems respectively for our results.

To describe our constructions in detail, we introduce a natural concept, the **weighted pseudorandom reduction**, which is defined as follows.

Definition 1.11 (Weighted Pseudorandom Reduction)**.** *Let $\mathcal{F}_0$ be a class of functions $(\{0,1\}^{s_0})^{n_0} \to \mathbb{R}$. Let $\mathcal{F}_{simp}$ be a class of functions $(\{0,1\}^{s_1})^{n_1} \to \mathbb{R}$. A (d, K, ε)-weighted pseudorandom reduction from $\mathcal{F}_0$ to $\mathcal{F}_{simp}$ is a tuple (R, w), in which $\mathsf{R} : (\{0,1\}^{s_1})^{n_1} \times \{0,1\}^d \to (\{0,1\}^{s_0})^{n_0}$ and $w : \{0,1\}^d \to \mathbb{R}$. Furthermore, for every $f \in \mathcal{F}_0$, we have:*

$$\left| \mathbb{E}_U f(U) - \frac{1}{2^d} \sum_{i \in \{0,1\}^d} w(i) \mathbb{E}_{U'}[f(\mathsf{R}(U', i))] \right| \le \varepsilon,$$
$$\forall i, |w(i)| \le K,$$
$$\forall i, f(\mathsf{R}(\cdot, i)) \in \mathcal{F}_{simp}.$$

We call R the reduction function and w the weight function. In some cases, we use the notation $\mathsf{R}_i(\cdot) := \mathsf{R}(\cdot, i)$ for convenience.

A weighted pseudorandom reduction can be viewed as a 'partial assignment' for the seed of a weighted pseudorandom generator. Instead of directly approximating $\mathbb{E}[f]$ with $\sum_r \frac{1}{2^s} w(r) \cdot f(r)$, we consider approximating $\mathbb{E}[f]$ with $\sum_i \frac{1}{2^d} w(i) \cdot \mathbb{E}_{U_{s_1}}[f(\mathsf{R}_i(U_{s_1}))]$. So for each i, we only need to further approximate $\mathbb{E}[f(\mathsf{R}_i(U_{s_1}))]$ with another weighted pseudorandom reduction. Despite its resemblance to error reduction frameworks, the weighted pseudorandom reduction is not designed for recursively reducing the error. Instead, the core idea is to approximate a complex ROBP by a weighted sum of simpler ROBPs (e.g. shorter or smaller alphabet), and recursively repeat this process until the ROBPs are simple enough to be handled easily.

WPRGs for standard-order ROBPs To describe our construction, we start by reviewing Armoni's generator in view of weighted pseudorandom reductions. For simplicity, we consider $\varepsilon \ge 1/w^3$. Armoni's generator can be viewed as a recursive weighted pseudorandom reduction procedure. It has k levels of recursions. Let f be the original ROBP. At level i of the recursion, assume the current ROBP is f_i of length n and alphabet $\{0,1\}^{C \log w}$, where $C > 0$ is a constant. We prepare a source X of $O(\log(nw/\varepsilon))$ bits and extracts it using multiple independent seeds $Y_1, \ldots, Y_n$, each of $O(\log(n/\varepsilon))$ bits. This outputs n chunks of $(C \log w)$-bit characters. And the number of random bits used is reduced from $Cn \log w$ to $O(n \log(n/\varepsilon))$. Notice that this is also a one-level NZ generator. If we further fix the source X, the composition of f_i and the one-level NZ generator can be viewed as a new ROBP that operates on $Y_1, \ldots, Y_n$, which can be further regarded as a ROBP of length $m = n \cdot \frac{O(\log(n/\varepsilon))}{C \log w}$ and alphabet $\{0,1\}^{C \log w}$. This perspective allows one to recursively apply the process. At each level i of recursion, one can reduce a ROBP of length n_i and alphabet $\{0,1\}^{C \log w}$ to a ROBP of length $n_{i+1} = n_i \cdot \frac{O(\log(n/\varepsilon))}{C \log w}$ and the alphabet $\{0,1\}^{C \log w}$. The cost of this level is that we use $O(\log(nw/\varepsilon))$ fresh random bits as a source to extract. By k levels of recursion, Armoni's generator achieves a seed length of $O\left(k \cdot \log(nw/\varepsilon) + n \cdot \left(\frac{\log(n/\varepsilon)}{\log w}\right)^k\right)$ for any integer $k \ge 1$. However, when $\varepsilon = 1/\operatorname{poly}(w)$, the multiplicative factor $\frac{O(\log(n/\varepsilon))}{C \log w}$ deteriorates to a constant. Under such cases, one can set k to be $\log n$ to attain a short seed. But then the seed length is no better than just applying Nisan's generator.

The first ingredient in our construction is the **length reduction**, which can accelerate the length-decreasing rate in the above procedure. For simplicity, we first focus on $n = \operatorname{poly} \log w$. We will handle larger n later. We show how to make $n_{i+1} = n_i^{1/3}$ and how to attain an advantage in seed

length based on this property. By a standard method from [CDR+21, PV21] based on Richardson iteration, one can transform a PRG with a moderate error τ into a WPRG with a tiny error ε. For level i, we set the moderate error $\tau = 2^{-\frac{C \log w}{n^{1/c}}}$ for some constants $C, c > 0$. The WPRG constructed in this manner has the form:

$$G(j, x_1, \ldots, x_m) = \mathsf{PRG}(x_1)_{n_{j,1}}, \mathsf{PRG}(x_2)_{n_{j,2}-n_{j,1}}, \ldots, \mathsf{PRG}(x_m)_{n_{j,m}-n_{j,m-1}},$$
$$w(j) = \sigma_j \cdot K.$$

Here $m = O\left(\frac{\log(n/\varepsilon)}{\log(1/\tau)}\right)$, j is chosen from a predefined set $[K]$, $K = n^{O(m)}$, $0 \le n_{j,1} < n_{j,2} < \ldots < n_{j,m} \le n$ are breakpoints, and $\sigma_j \in \{-1, 0, 1\}$. All these parameters are from a certain preconditioned Richardson iteration. The PRG $\mathsf{PRG}(\cdot)$ that we use here, is the NZ generator with c levels of recursion, maintaining a seed length of $O(\log w)$. The output sequence is a concatenation of m chunks of independent NZ generator outputs, each truncated to a length of $n_{j,t} - n_{j,t-1}$ bits. To repeat the process, we convert the WPRG into a weighted pseudorandom reduction by fixing j and allowing $x_1, \ldots, x_m$ to remain free. Namely, we define (R, w), with $\mathsf{R}_j(x) = \mathsf{PRG}(x_1)_{n_{j,1}}, \mathsf{PRG}(x_2)_{n_{j,2}-n_{j,1}}, \ldots, \mathsf{PRG}(x_m)_{n_{j,m}-n_{j,m-1}}$ and $w(j) = \sigma_j \cdot K$. The reduced ROBP can be viewed as running on m large characters, each corresponding to a seed of an NZ generator $\mathsf{PRG}(\cdot)$. The key point is that by the standard calculation in [CDR+21, PV21], $m = O\left(\frac{\log(n/\varepsilon)}{\log(1/\tau)}\right) = n^{1/c}$, if $\varepsilon = 1/\operatorname{poly} w$ and we take the constant C in τ to be large enough. Hence if we take $c = 3$, $n = n_i$, then $n_{i+1} = n_i^{1/3}$.

In principle, this process can be repeated until n_i is reduced to a constant, requiring $l = \log\log n = \log\log\log w$ levels of recursion. However, when we apply the NZ generator for a ROBP with a large alphabet $\{0,1\}^s$, it requires a seed of $\Theta(s + \log w)$ bits, the constant hidden in big-Θ can at least double the bit-length of the alphabet. Hence such a direct repetition may lead to a final alphabet of $\log w \cdot 2^l = O(\log w \log\log w)$ bits, which loses the advantage in seed length.

To solve this problem, we introduce another reduction, the **alphabet reduction**, which can control the size of the alphabet using a one-level NZ generator. Assume we have a ROBP f with length m and alphabet log-size being some large s. Recall that a one-level NZ generator, on an input source X of $O(s + \log w)$ bits, and a sequence of independent seeds $Y_1, \ldots, Y_m$, each of $d = O(\log(n/\varepsilon))$ bits, can output m characters each of length s. So specifically, we can define (R, w), where $\mathsf{R}_x(y_1, y_2, \ldots, y_m) = \textsc{Ext}(x, y_1), \textsc{Ext}(x, y_2), \ldots, \textsc{Ext}(x, y_m)$ and $w(x) \equiv 1$. Hence by fixing $X = x$, then $f(\mathsf{R}_x)(\cdot)$ is a ROBP with length m and alphabet log-size d which can be a small constant fraction times $|x|$.

Now we describe our whole recursive reduction that alternates between length reductions and alphabet reductions. Let $\varepsilon = 1/\operatorname{poly}(w)$. We will handle large n later, but now assume at some step we have already obtained ROBPs of length $n_i \le \log^2 w$, alphabet log-size $s_i = C_1 \log w$, and width w. Next, we first apply the length reduction, which gives us ROBPs of length $n_{i+1} = n_i^{1/3}$, width remaining the same, and alphabet log-size $O(s_i) = C \log w$ for some large constant C. The weight of this reduction is $K = n_i^{O(n_{i+1})} \le 2^{O(n_{i+1} \log n_i)}$, and we need $\log K$ random bits to choose an addend. Then we apply the alphabet reduction, which gives us ROBPs of the same length n_{i+1}, width w, but alphabet log-size $s_{i+1} = O(\log(n_{i+1}/\varepsilon)) = C_1 \log w = s_i$. This reduction requires $O(\log w)$ bits of randomness as a source for the extractor. Combining these two steps, we reduce n_i to n_{i+1} and maintain the same width and alphabet. The overall cost is $O(\log w)$ bits of randomness and weight $W \le 2^{O(n_{i+1} \log n_i)}$. One can repeat this process until n is small enough, and this gives our new WPRG. A detailed calculation can deduce that the whole process needs randomness $l \cdot O(\log w) = O(\log w \log\log\log w)$. The overall weight W is the product of weights from each level,

i.e. $W = 2^{\sum_i O(n_{i+1}\log n_i)} << \text{poly}\, w$. Also, notice that one can let ε be our target error times $1/W$ to bound the overall error.

As mentioned before, the above approach works for a ROBP of length $n = \log^2 w$. To extend it to a large length, we use an extra length reduction based on Armoni's PRG and the standard error reduction [CDR+21, PV21], to reduce the original ROBP to a ROBP that has length $<< O(\log^2 w)$, which fits into the regime we have already settled.

Furthermore, to achieve a smaller target error $\varepsilon << 1/\text{poly}(w)$. We slightly modify the sampler technique given by Hoza [Hoz21], to work on our above construction. We mention that this operation can also be viewed as an extra level of weighted pseudorandom reduction. This ultimately gives the WPRG in Theorem 1.5.

WPRGs for unbounded-width permutation ROBPs with a single accept node We start by reviewing the recent WPRG [CHL+23] in a weighted pseudorandom reduction view. Their WPRG combines the INW generator and a new matrix iteration. To fool a unbounded-width permutation ROBP of length n and alphabet $\{0,1\}^s$ within error ε, they first prepare a INW generator $\mathsf{PRG} : \{0,1\}^d \to (\{0,1\}^s)^n$ with moderate sv-error[3] $\tau = 2^{O(\log\log n\sqrt{\log(1/\varepsilon)})}$. Then they construct the ε-WPRG (G, w) in the following form:

$$G(j, x_1, \ldots, x_m) = \mathsf{PRG}(x_1)_{n_{j,1}}, \mathsf{PRG}(x_2)_{n_{j,2}-n_{j,1}}, \ldots, \mathsf{PRG}(x_m)_{n_{j,m}-n_{j,m-1}},$$
$$w(j) = \sigma_j \cdot K.$$

Here $m = O\left(\log n \cdot \frac{\log(\log n/\varepsilon)}{\log(1/\tau)}\right)$, j is choosed from set $[K]$, $K = 2^{O(m)}$, $0 \le n_{j,1} < n_{j,2} < \ldots < n_{j,m} \le n$ are the breakpoints, and $\sigma_j \in \{-1, 0, 1\}$. All these parameters are from their new matrix iteration. The seed length of such INW generator is $d = s + O\left(\log n \cdot (\log\log n + \log(1/\tau))\right)$. Denote (G, w) with the above parameters as $\mathsf{R}^{(0)}$. Viewing $\mathsf{R}^{(0)}$ as a weighted pseudorandom reduction, it reduces the original ROBP to ROBPs of length m and alphabet $\{0,1\}^d$. Finally, they use an ε/K-INW generator to fool these reduced permutation ROBPs, which cost $d + O\left(\log m \cdot (\log\log m + \log(K/\varepsilon)\right)$ bits of randomness. This step introduces some double-logarithmic factors.

We improve the construction, eliminating the double-logarithmic factors in the seed length, by replacing the last INW generator with recursive weighted pseudorandom reductions. We start from the reduced ROBP of length n_1 and alphabet $\{0,1\}^{s_1}$, where $n_1 = m$ and $s_1 = d$. We set $\varepsilon' = (\varepsilon/(2K))^2$ as the target error for upcoming weighted reductions. At the ith level of the recursion, assume the current permutation ROBP has length n_i and alphabet $\{0,1\}^{s_i}$. We use the above WPRG (G, w) to fool it except that we set the target error to ε' and the moderate error to $\tau = 2^{\frac{C\log(1/\varepsilon')}{\log^2 n_i}}$, where $C > 0$ is a constant. Denote (G, w) with these parameters as $\mathsf{R}^{(i)}$. Then we can view $\mathsf{R}^{(i)}$ as a weighted pseudorandom reduction. It can reduce the current permutation ROBP to permutation ROBPs of length n_{i+1} and alphabet $\{0,1\}^{s_{i+1}}$. Here $n_{i+1} = O\left(\log n_i \frac{\log(1/\varepsilon')}{\log(1/\tau)}\right) = \log^3 n_i$ if we set C in τ to be large enough. And s_{i+1} is identical to the seed length of the INW generator $\mathsf{PRG}(\cdot)$ used in $\mathsf{R}^{(i)}$, thus $s_{i+1} = s_i + O(\log n_i \cdot (\log\log n_i + \log(1/\tau))) = s_i + O\left(\frac{\log(1/\varepsilon')}{\log n_i}\right)$. We emphasize that the increment from s_i to s_{i+1} is not significant, hence we do not need an alphabet reduction to control the alphabet size. This level of recursion has weight $W_i = 2^{O(\log^3 n_i)}$ and costs $\log W_i$ bits of randomness. We do the recursion for $l-1$ times, where l is chosen in a way such that $n_l = O(1)$. The total weight of the recursion is bound by $2^{O(\log^3 n_1)}$ since n_i decreases quickly. Notice that this is negligible compared to ε/K, which means that $\varepsilon' = (\varepsilon/(2K))^2$ is small enough to deduce an ε-WPRG. Finally, we calculate

[3] See Definition 4.5.

the random bits used. The overall random bits used in our WPRG have three parts, the bits for the final ROBP of length n_l and alphabet $\{0,1\}^{s_l}$, the bits used in the recursion, and the bits used in the first weighted reduction to choose an addend of the matrix iteration polynomial. For the first part, we have $s_l = s_1 + O(\log(1/\varepsilon')) \cdot \sum_{i=1}^{l-1} \frac{1}{\log n_i} = s_1 + O(\log(1/\varepsilon'))$. For the second part, one can deduce that the number of bits is $\leq O(\log^3 n_1)$, since $foralli, n_{i+1} = \log^3 n_i$. This is negligible compared to $\log(1/\varepsilon')$. The third part requires $\log K$ bits. Therefore, the overall random bits used in our WPRG is $s_l \cdot n_l + O(\log^3 n_1) + \log W = O(s_1 + \log(1/\varepsilon') + \log K) = O(s + \log n \cdot (\log\log n + \sqrt{\log(1/\varepsilon)}) + \log(1/\varepsilon))$.

Derandomization of short-wide regular ROBPs Our derandomization adapts our WPRG for permutation ROBPs and utilizes the rotation technique which is introduced in [RVW00, RV05] and commonly used in [AKM+20, CHL+23, CL24]. Recall that the rotation technique in [AKM+20, CHL+23, CL24] is used to replace the expander walks in INW and also those one-step transitions on the input ROBP in a white-box way, such that one can use this adapted WPRG to fool the regular ROBP as if one is using the original WPRG to fool a permutation ROBP. Although our WPRG is constructed by recursive weighted pseudorandom reductions, notice that for each level of the recursion, it still inherits the expander walk structure of INW. We show that one can use a sequence of different rotations for different levels of the recursion and different positions of the ROBP to adapt the WPRG such that it can simulate random walks on the input regular ROBP. The parameters basically remain the same as that of our WPRG for permutation ROBPs.

2 Preliminaries

2.1 Some notations

For convenience of description, we denote $\mathcal{B}_{(n,s,w)}$ as the class of ROBPs with length n, alphabet size 2^s and width w. We denote $\mathcal{P}_{(n,s)}$ as the class of permutation ROBPs with unbounded width and only one accept node, where n is the length s is the log-size of the alphabet. We denote $\mathcal{R}_{(n,s,w)}$ as the class of regular ROBPs with length n, alphabet size 2^s and width w.

Given an ROBP f, we denote $f^{[i,j]} : (\{0,1\}^s)^{j-i} \to \mathbb{R}^{w \times w}$ as the matrix representing the transition function of f between layers V_i and V_j. Specifically, the entry $[f^{[i,j]}(x)]_{u,v} = 1$ if there exists a path from node u in V_i to node v in V_j that is labeled by the string $x = x_1 \dots x_{j-i}$. Otherwise, $[f^{[i,j]}(x)]_{u,v} = 0$.

We use $\circ$ to denote the composition of functions, i.e. for any two functions f, g where any output of g can be an input of f, we have that $f \circ g(x) := f(g(x))$.

2.2 Weighted pseudorandom reductions

Here we recall the definition of Weighted Pseudorandom Reduction.

Definition 2.1 (Weighted Pseudorandom Reduction)**.** *Let $\mathcal{F}_0$ be a class of functions $(\{0,1\}^{s_0})^{n_0} \to \mathbb{R}$. Let $\mathcal{F}_{simp}$ be a class of functions $(\{0,1\}^{s_1})^{n_1} \to \mathbb{R}$. A (d, K, ε)-weighted pseudorandom reduction from $\mathcal{F}_0$ to $\mathcal{F}_{simp}$ is a tuple (R, w), in which $\mathsf{R} : (\{0,1\}^{s_1})^{n_1} \times \{0,1\}^d \to (\{0,1\}^{s_0})^{n_0}$ and w :*

$\{0,1\}^d \to \mathbb{R}$. *Furthermore, for every* $f \in \mathcal{F}_0$, *we have:*

$$\left|\mathbb{E}_U f(U) - \frac{1}{2^d}\sum_{i\in\{0,1\}^d} w(i)\mathbb{E}_{U'}[f(\mathsf{R}(U',i))]\right| \le \varepsilon,$$
$$\forall i, |w(i)| \le K,$$
$$\forall i, f(\mathsf{R}(\cdot,i)) \in \mathcal{F}_{simp}.$$

We call R *the reduction function and* w *the weight function. In some cases, we use the notation* $\mathsf{R}_i(\cdot) := \mathsf{R}(\cdot,i)$ *for convenience.*

We emphasize that throughout this work, both $\mathcal{F}_0$ and $\mathcal{F}_{simp}$ are some classes of ROBPs, i.e. all our reductions keep this read-once property which is crucial in our proof.

We will frequently use compositions of reductions.

Lemma 2.2 (Composition Lemma)**.** *Let* $\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2$ *be classes of boolean functions within* $\{0,1\}^{s_0} \to \mathbb{R}, \{0,1\}^{s_1} \to \mathbb{R}, \{0,1\}^{s_2} \to \mathbb{R}$ *respectively. Let* $(\mathsf{R}^{(1)}, w^{(1)})$ *be an explicit* $(d_1, K_1, \varepsilon_1)$*-weighted pseudorandom reduction from* $\mathcal{F}_0$ *to* $\mathcal{F}_1$ *and* $(\mathsf{R}^{(2)}, w^{(2)})$ *be an explicit* $(d_2, K_2, \varepsilon_2)$*-weighted pseudorandom reduction from* $\mathcal{F}_1$ *to* $\mathcal{F}_2$*. Then the composition* $(\mathsf{R}^{(1)} \circ \mathsf{R}^{(2)}, w^{(1)} \cdot w^{(2)})$:

$$(\mathsf{R}^{(1)} \circ \mathsf{R}^{(2)})_{(i_1,i_2)}(x) := \mathsf{R}^{(1)}_{i_1}(\mathsf{R}^{(2)}_{i_2}(x)),$$
$$w^{(1)} \cdot w^{(2)}(i_1,i_2) := w^{(1)}(i_1) \cdot w^{(2)}(i_2),$$

is an explicit $(d_1+d_2, K_1K_2, \varepsilon_1 + K_1\varepsilon_2)$*-weighted pseudorandom reduction from* $\mathcal{F}_0$ *to* $\mathcal{F}_2$*.*

Proof. Let $\mathsf{R} = \mathsf{R}^{(1)} \circ \mathsf{R}^{(2)}$ and $w = w^{(1)} \cdot w^{(2)}$. For any $f \in \mathcal{F}_0$, we have:

$$\begin{aligned}
&\left|\mathbb{E}_U f(U) - \frac{1}{2^{d_1+d_2}}\sum_{i_1\in\{0,1\}^{d_1}, i_2\in\{0,1\}^{d_2}} w(i_1,i_2)\mathbb{E}_{U_{s_2}}\left[f(\mathsf{R}_{(i_1,i_2)}(U_{s_2}))\right]\right| \\
&\le \left|\mathbb{E}_U f(U) - \frac{1}{2^{d_1}}\sum_{i_1\in\{0,1\}^{d_1}} w^{(1)}(i_1)\mathbb{E}_{U_{s_1}}\left[f(\mathsf{R}^{(1)}_{i_1}(U_{s_1}))\right]\right| \\
&+ \frac{1}{2^{d_1}}\sum_{i_1\in\{0,1\}^{d_1}} |w^{(1)}(i_1)|\left|\mathbb{E}_{U_{s_1}}\left[f(\mathsf{R}^{(1)}_{i_1}(U_{s_1}))\right] - \frac{1}{2^{d_2}}\sum_{i_2\in\{0,1\}^{d_2}} w^{(2)}(i_2)\mathbb{E}_{U_{s_2}}\left[f(\mathsf{R}^{(2)}_{i_2}(U_{s_2}))\right]\right| \\
&\le \varepsilon_1 + K_1\varepsilon_2.
\end{aligned}$$

Furthermore, $|w(i_1,i_2)| \le |w^{(1)}(i_1)| \cdot |w^{(2)}(i_2)| \le K_1K_2$. For any i_1, i_2, $x \mapsto f(\mathsf{R}^{(1)}_{i_1}(x)) \in \mathcal{F}_1$, sp we can apply $\mathsf{R}^{(2)}_{i_2}$ to this mapping, therefore $x \mapsto f(\mathsf{R}_{(i_1,i_2)}(x)) \in \mathcal{F}_2$. The seed length is d_1+d_2. The function is explicit since both components are explicit. □

Lemma 2.3 (Composition Lemma for multiple reductions)**.** *For any positive integer* k*, let* $\mathcal{F}_0, \mathcal{F}_1, \ldots, \mathcal{F}_k$ *be classes of boolean functions within* $\{0,1\}^{s_0} \to \mathbb{R}, \{0,1\}^{s_1} \to \mathbb{R}, \ldots, \{0,1\}^{s_k} \to \mathbb{R}$ *respectively. Let* $(\mathsf{R}^{(1)}, w^{(1)}), \ldots, (\mathsf{R}^{(k)}, w^{(k)})$ *be explicit* $(d_1, K_1, \varepsilon_1), \ldots, (d_k, K_k, \varepsilon_k)$*-weighted pseudorandom reductions from* $\mathcal{F}_0$ *to* $\mathcal{F}_1, \ldots, \mathcal{F}_k$ *respectively. Then the composition* $(\mathsf{R}^{(1)} \circ \ldots \circ \mathsf{R}^{(k)}, w^{(1)} \cdot \ldots \cdot w^{(k)})$:

$$(\mathsf{R}^{(1)} \circ \ldots \circ \mathsf{R}^{(k)})_{(i_1,\ldots,i_k)}(x) = \mathsf{R}^{(1)}_{i_1}(\ldots \mathsf{R}^{(k)}_{i_k}(x)\ldots),$$
$$w^{(1)} \cdot \ldots \cdot w^{(k)}(i_1,\ldots,i_k) = w^{(1)}(i_1) \cdot \ldots \cdot w^{(k)}(i_k),$$

is an explicit $(\sum_{i=1}^{k} d_i, \prod_{i=1}^{k} K_i, \sum_{i=1}^{k} \left(\prod_{j=1}^{i-1} K_j\right) \varepsilon_i)$*-weighted pseudorandom reduction from* $\mathcal{F}_0$ *to* $\mathcal{F}_k$*.*

Proof. We prove this by induction on k. The base case $k = 2$ is shown by Lemma 2.2. Suppose the statement holds for $k-1$, then $(\mathsf{R}^{(1)} \circ \ldots \circ \mathsf{R}^{(k-1)}, w^{(1)} \cdot \ldots \cdot w^{(k-1)})$ is an explicit $(\sum_{i=1}^{k-1} d_i, \prod_{i=1}^{k-1} K_i, \sum_{i=1}^{k-1} \left(\prod_{j=1}^{i-1} K_j\right) \varepsilon_i)$-weighted pseudorandom reduction from $\mathcal{F}_0$ to $\mathcal{F}_{k-1}$. Then we can apply Lemma 2.2 on $(\mathsf{R}^{(1)} \circ \ldots \circ \mathsf{R}^{(k-1)}, w^{(1)} \cdot \ldots \cdot w^{(k-1)})$ and $(\mathsf{R}^{(k)}, w^{(k)})$, which gives the desired result. □

3 WPRG for Standard ROBPs

In this section, we provide a new construction of Weighted Pseudorandom Generator(WPRG) for read-once branching programs(ROBPs). The main idea is iteratively approximating the expectation of a long ROBP by a weighted sum of the expectations of much shorter ROBPs. Let f be a long ROBP, our reduction will follow the paradigm:

$$\left| \mathbb{E} f(U) - \frac{1}{2^d} \sum_{i \in 2^d} w^{(i)} \mathbb{E}\left[f(\mathsf{R}^{(i)}(U)) \right] \right| < \varepsilon,$$

where $f \circ \mathsf{R}^{(i)}$ could be computed by a much shorter ROBP for any fixed i and f.

We mainly use two types of weighted pseudorandom reductions: alphabet reduction and length reduction.

Alphabet Reduction: In an alphabet reduction (R, w), each $\mathsf{R}^{(i)}$ is a $(\{0,1\}^{s'})^n \to (\{0,1\}^s)^n$ function that maps n short characters to n long characters. For any $f \in \mathcal{B}_{(n,s,w)}$, $f \circ \mathsf{R}^{(i)}$ could be computed in $\mathcal{B}_{(n,s',w)}$, where the length and width are unchanged but the alphabet log-size is reduced.

Length Reduction: In a length reduction (R, w), each $\mathsf{R}^{(i)}$ is a $(\{0,1\}^{s'})^k \to (\{0,1\}^s)^n$ function that maps k long characters to $n \gg k$ short characters. For any $f \in \mathcal{B}_{(n,s,w)}$, $f \circ \mathsf{R}^{(i)}$ could be computed in $\mathcal{B}_{(k,s',w)}$, where the length is reduced from n to k but the alphabet log-size is increased to s'.

For the rest of this section, we are going to show the following lemma. The general strategy is to apply the two reductions alternately to reduce the length and maintain the width and alphabet log-size.

Lemma 3.1. *For every* $\varepsilon \geq 1/\operatorname{poly}(w)$*, there exists an explicit* ε*-WPRG for* $\mathcal{B}_{(n,s,w)}$ *with seed length* $O\left(s + \frac{\log n \log(nw)}{\max\{1, \log\log w - \log\log n\}} + \log w(\log\log\log w - \log\log \frac{\log w}{\log n/\varepsilon})\right)$ *and weight* $(8n)^{2\sqrt{\frac{\log w}{\log n}}}$*.*

We note that for smaller target errors, one can further use the sampler trick [Hoz21] to reduce the error from $1/\operatorname{poly}(w)$ to an arbitrary $\varepsilon > 0$, attaining the following theorem.

Theorem 3.2. *For all integer* n, s, w*, there exists an explicit construction of a* ε*-WPRG for* $\mathcal{B}_{(n,s,w)}$ *with seed length*

$$O\left(s + \frac{\log n \log(nw)}{\max\{1, \log\log w - \log\log n\}} + \log w \left(\log\log\log w - \log\log \max\left\{2, \frac{\log w}{\log n/\varepsilon}\right\}\right) + \log(1/\varepsilon)\right)$$

Note that our main theorem Theorem 1.5 is a direct corollary of Theorem 3.2.

3.1 Alphabet Reduction

In this section, we will provide a construction of the alphabet reduction. The alphabet reduction reduces the alphabet log-size s to a much smaller $O(\log w)$ and keeps the length n and width w unchanged. The alphabet reduction is achieved by using the Nisan-Zuckerman (NZ) PRG, we start with the construction of the PRG and its main ingredient, extractors.

Definition 3.3 (Extractor)**.** *A (k, ε)-extractor is a function $\textsc{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^k$ such that for any $X \in \{0,1\}^n$ with $H(X) \geq k$, the distribution of $\textsc{Ext}(X, U_d)$ is ε-close to the uniform distribution on $\{0,1\}^k$. Here $H(X) = \max_{x \in \{0,1\}^n} -\log \Pr[X = x]$ is the min-entropy of X, and U_d is the uniform distribution on $\{0,1\}^d$.*

Theorem 3.4 (Explicit Extractor from [GUV09])**.** *There exists a universal constant C_{GUV} that for all positive integer s and positive real ε, there is an explicit construction of a $(2s, \varepsilon/3)$-extractor $\textsc{Ext} : \{0,1\}^{3s} \times \{0,1\}^d \to \{0,1\}^s$ with seed length $d = C_{GUV} \cdot \log \frac{ns}{\varepsilon}$.*

We mainly use the one-level version of NZ PRG.

Lemma 3.5 (One-level NZ PRG[NZ96] with a large alphabet)**.** *Let $s \geq \log w$. Assume there exists a $(2s, \frac{\varepsilon}{3n})$-extractor $\textsc{Ext} : \{0,1\}^{3s} \times \{0,1\}^d \to \{0,1\}^s$. Let X and $Y_1, \ldots, Y_n$ be independent uniform random variables. Then the following construction*

$$\mathsf{NZ}(X, Y) = \textsc{Ext}(X, Y_1), \ldots, \textsc{Ext}(X, Y_n)$$

fools any $f \in \mathcal{B}_{(n,s,w)}$ with error at most ε.

The original proof of [NZ96] also works for large alphabet. We include a proof in Appendix A for completeness.

Now we construct the alphabet reduction. Given a ROBP $f \in \mathcal{B}_{(n,s,w)}$, we use $\mathbb{E} f(\mathsf{NZ}(X, Y))$ to approximate $\mathbb{E} f$. By fixing X, the computation of $f(\mathsf{NZ}(X, Y))$ can be done by a program of much smaller alphabet, that gives the following reduction.

Lemma 3.6 (Alphabet Reduction)**.** *For all positive integers w, n, s with $s \geq \log w$, there exists an explicit $(3s, 1, \varepsilon)$-weighted pseudorandom reduction from $\mathcal{B}_{(n,s,w)}$ to $\mathcal{B}_{(n, C_{GUV} \log \frac{ns}{\varepsilon}, w)}$.*

Proof. Let $\textsc{Ext} : \{0,1\}^{3s} \times \{0,1\}^d \to \{0,1\}^s$ be a $(2s, \frac{\varepsilon}{3n})$-extractor with seed length $d = C_{GUV} \cdot \log \frac{ns}{\varepsilon}$ from Theorem 3.4. Let NZ be defined as in Lemma 3.5. We define the reduction (R, w):

$$\mathsf{R}_x(y) := \mathsf{NZ}(x, y)$$
$$w_x := 1.$$

Now consider that we fix x and let y_i's be free. We need to prove that the reduction (R, w) is a $(3s, 1, \varepsilon)$-weighted pseudorandom reduction.

Let $f \in \mathcal{B}_{(n,s,w)}$, then by Lemma 3.5,

$$|\mathbb{E} f - 2^{-3s} \sum_{x \in \{0,1\}^{3s}} \mathbb{E} f(\mathsf{NZ}(x, *))| = |\mathbb{E} f - \mathbb{E} f(\mathsf{NZ}(X, Y))| \leq \varepsilon.$$

Therefore, (R, w) approximates f with error at most ε.

To prove that $f(\mathsf{NZ}(x, *)) \in \mathcal{B}_{(n,d,w)}$ for all $f \in \mathcal{B}_{(n,s,w)}$ and $x \in \{0,1\}^{3s}$, $y \in \{0,1\}^d$. We construct the ROBP $g_x := f(\mathsf{NZ}(x, *))$ as follows:

Let $V_0, \ldots, V_n$ be the layers of f, each consisting of w nodes. The ROBP g_x is also defined on $V_0, \ldots, V_n$. The labeled edge set of g_x is defined as follows:

$$E_{g_x} = \{(u, v, y) : u \in V_{i-1}, v \in V_i, y \in \{0,1\}^d, \exists \text{ an edge from } u \text{ to } v \text{ labeled by } \textsc{Ext}(x, y) \text{ in } f\}$$

The start node and accept nodes of G are the same as those in f. Then g_x is in $\mathcal{B}_{(n,d,w)}$ and $g_x(y_1, \ldots, y_n) = f(\textsc{Ext}(x, y_1), \ldots, \textsc{Ext}(x, y_n)) = f(\mathsf{NZ}(x, y))$. □

3.2 Length Reduction framework from Richardson Iteration

A length reduction reduces the length n to a much smaller k but may increase the alphabet log-size. In this subsection, we construct a length reduction using the error reduction given by [AKM+20, CDR+21, PV21], and some explicit PRGs against ROBPs.

Lemma 3.7 (framework of the length reduction)**.** *For any positive integers n, s, w, positive odd integer k and positive real ε, assume there exists an explicit $\frac{\varepsilon}{(n+1)^2}$-PRG for $\mathcal{B}_{(n,s,w)}$ with seed length $d = d(n, s, w)$. Then there exists an explicit $(\log K, K, \varepsilon^{\frac{k+1}{2}} \cdot (n+1))$-weighted pseudorandom reduction from $\mathcal{B}_{(n,s,w)}$ to $\mathcal{B}_{(k,d,w)}$, where $K = (8n)^{k+1}$.*

To prove Lemma 3.7, we need the following result.

Theorem 3.8 (Error Reduction based on Richardson Iteration [AKM+20][CDR+21][PV21])**.** *Let $\{A_i\}_{i=1}^n \subset \mathbb{R}^{w \times w}$ be a sequence of matrices. Let $\{B_{i,j}\}_{i,j=0}^n \subset \mathbb{R}^{w \times w}$ be a family of matrices such that for every $i+1 < j$, $\|B_{i,j} - A_{i+1} \ldots A_j\| \le \varepsilon/(n+1)^2$ for some submultiplicative norm $\|\cdot\|$, $\|A_i\| \le 1$ for all i and also $B_{i-1,i} = A_i$ for all i. Then for any odd $k \in \mathbb{N}$, there exists a $K = (8n)^{k+1}$, a set of indices $\{n_{i,j}\}_{i \in [K], j \in [k]}$ with $0 \le n_{i,1} \le \ldots \le n_{i,k} = n$, and signs $\sigma_i \in \{-1, 0, 1\}, i \in [K]$ such that (We set $B_{i,i} = I$ for all i):*

$$\left\| A - \sum_{i \in [K]} \sigma_i \cdot B_{0,n_{i,1}} B_{n_{i,1},n_{i,2}} \ldots B_{n_{i,k-1},n_{i,k}} \right\| \le \varepsilon^{(k+1)/2} \cdot (n+1).$$

A proof of Theorem 3.8 is in Appendix B.

Proof of Lemma 3.7. Let $k, K, n_{i,j}, \sigma_i$ be as in Theorem 3.8, Let PRG be a $\varepsilon/(n+1)^2$-PRG for $\mathcal{B}_{(n,s,w)}$ with seed length d in the assumption. We define the reduction (R, w) as follows:

$$\mathsf{R}_i(x_1, \ldots, x_k) = \mathsf{PRG}(x_1)_{n_{i,1}}, \mathsf{PRG}(x_2)_{n_{i,2}-n_{i,1}}, \ldots, \mathsf{PRG}(x_k)_{n_{i,k}-n_{i,k-1}}$$
$$w_i = \sigma_i K.$$

Here $\mathsf{PRG}(x_i)_c$ denotes the first c characters of $\mathsf{PRG}(x_i)$ and each character is in $\{0,1\}^s$.

We will show that (R, w) is a $(\log K, K, \varepsilon^k \cdot (n+1))$-weighted pseudorandom reduction from $\mathcal{B}_{(n,s,w)}$ to $\mathcal{B}_{(k,d,w)}$.

Let $f \in \mathcal{B}_{(n,s,w)}$. Recall that $f^{[i,j]}(x)$ denotes the transition matrix of f from layer i to layer j with input $x = (x_1, \ldots, x_{j-i})$. Define $A_i = \mathbb{E}_{x \in \{0,1\}^s} f^{[i-1,i]}[x]$ and $B_{i,j} = \mathbb{E}_{x \in \{0,1\}^d} f^{[i,j]}[\mathsf{PRG}(x)_{j-i}]$. By the definition of the PRG, we have $\|B_{i,j} - A_{i+1} \ldots A_j\|_1 \le \varepsilon/(n+1)$.

Therefore, by Theorem 3.8,

$$\begin{aligned}
&\left|\mathbb{E}f - \frac{1}{K}\sum_{i=1}^{K} w(i)\mathbb{E}_X f(\mathsf{R}_i(X))\right| \\
=&\left\|\mathbb{E}f^{[0,n]} - \sum_{i=1}^{K}\sigma_i \mathbb{E}f^{[0,n]}(\mathsf{PRG}(X_1)_{n_{i,1}}, \mathsf{PRG}(X_2)_{n_{i,2}-n_{i,1}}, \ldots, \mathsf{PRG}(X_k)_{n_{i,k}-n_{i,k-1}})\right\| \\
\leq&\left\|\mathbb{E}f^{[0,n]} - \sum_{i=1}^{K}\sigma_i \mathbb{E}f^{[0,n_{i,1}]}(\mathsf{PRG}(X_1)_{n_{i,1}})\mathbb{E}f^{[n_{i,1},n_{i,2}]}(\mathsf{PRG}(X_2)_{n_{i,2}-n_{i,1}})\cdots\right. \\
&\qquad\left.\cdots\mathbb{E}f^{[n_{i,k-1},n_{i,k}]}(\mathsf{PRG}(X_k)_{n_{i,k}-n_{i,k-1}})\right\| \\
=&\left\|A_1\ldots A_n - \sum_{i=1}^{K}\sigma_i B_{0,n_{i,1}}B_{n_{i,1},n_{i,2}}\ldots B_{n_{i,k-1},n_{i,k}}\right\| \\
\leq&\varepsilon^{\frac{k+1}{2}}\cdot(n+1).
\end{aligned}$$

The weight is $K = (8n)^{k+1}$ by Theorem 3.8, and so the seed length is $\log K = O(k\log n)$.

Finally, we need to show that $f \circ \mathsf{R}_i \in \mathcal{B}_{(k,d,w)}$ for all $f \in \mathcal{B}_{(n,s,w)}$ and $i \in [K]$. We construct the ROBP $g := f \circ \mathsf{R}_i$ as follows:

Let $V_0, \ldots, V_n$ be the layers of f, each consisting of w nodes. The ROBP g is defined on $V_0, V_{n_{i,1}}, \ldots, V_{n_{i,k}}$. The labelled edge set of g is defined as follows:

$$\begin{aligned}
E_g = \{&(u,v,x) : j \in [n], u \in V_{n_{i,j-1}}, v \in V_{n_{i,j}}, x \in \{0,1\}^d, \\
&\text{there exists a path from } u \text{ to } v \text{ in } f \text{ through } \mathsf{PRG}(x)_{n_{i,j}-n_{i,j-1}}\ \}
\end{aligned}$$

The start node and accept nodes of g are the same as that in f. Then g is in $\mathcal{B}_{(k,d,w)}$ and $g(x_1,\ldots,x_n) = f(\mathsf{PRG}(x_1)_{n_{i,1}}, \mathsf{PRG}(x_2)_{n_{i,2}-n_{i,1}}, \ldots, \mathsf{PRG}(x_k)_{n_{i,k}-n_{i,k-1}}) = f(\mathsf{R}_i(x_1,\ldots,x_k))$. □

3.3 Length Reduction instantiated by Armoni's PRG

We use Armoni's PRG for $\mathcal{B}_{(n,s,w)}$ to instantiate the framework of Section 3.2.

Theorem 3.9 (Armoni's PRG[Arm98, KNW08])**.** *For all positive integer n, s, w and positive real ε, there exists an explicit construction of a ε-PRG for $\mathcal{B}_{(n,s,w)}$ with seed length $O(s + \frac{\log n \log(nw/\varepsilon)}{\log\log w - \log\log(n/\varepsilon)})$. Here the big-O of the seed length hides a universal constant.*

We have two instantiations that have different parameters.

Lemma 3.10 (Length reduction 1)**.** *For any constant $a, c, C \in \mathbb{N}$, for all integer n, w, if $n \leq \log^c w$, then there exists an explicit $(\log K, K, 1/w^a)$-weighted pseudorandom reduction from $\mathcal{B}_{(n,C\log w,w)}$ to $\mathcal{B}_{(n^{1/c},C'\log w,w)}$, where $K \leq (8n)^{n^{1/c}+1}$, C' is a constant depending on c, C, a.*

Proof. Fix c, C to be constants. We use Armoni's PRG[4] for $\mathcal{B}_{(n,C\log w,w)}$ with error $\varepsilon_0 = \frac{2^{-\frac{a\log w}{n^{1/c}}}}{(n+1)^2}$. By Theorem 3.9, the seed length is $O\left(C\log w + \frac{a\log w\log\log w}{\log\log w/c + O(1)}\right) = C'\log w$, where C' depends on c, C, a. Then we can set $k = 2n^{1/c} \geq \frac{a\log w + \log n}{1/2\cdot\log(1/\varepsilon_0)}$ in Lemma 3.7 and invoke it to attain this lemma. □

[4]Readers can notice that for this setting, the Armoni's PRG that we use, should automatically become the Nisan-Zuckerman PRG.

Lemma 3.11 (Length reduction 2)**.** *For any constant $a \in \mathbb{N}$, for all integer n, s, w, there exists an explicit $(\log K, K, 1/w^a)$-weighted pseudorandom reduction from $\mathcal{B}_{(n,s,w)}$ to $\mathcal{B}_{\left(\log^{\frac{1}{2}} w, O\left(s+\frac{\log n \log(nw)}{\log\log w - \log\log n}\right), w\right)}$, where $K = (8n)^{\sqrt{\frac{\log w}{\log n}}+1}$.*

Proof. We use Armoni's PRG for $\mathcal{B}_{(n,s,w)}$ with error $\varepsilon_0 = \frac{2^{-2a\sqrt{\log n \log w}}}{(n+1)}$. By Theorem 3.9, the seed length is $O\left(s + \frac{\log n \log(nw)}{\log\log w - \log\log n}\right)$. Then we can set $k = \frac{a \log w + \log n}{1/2 \cdot \log(1/\varepsilon_0)} \leq \sqrt{\frac{\log w}{\log n}} \leq \log^{1/2} w$ in Lemma 3.7 and invoke it to attain this lemma. □

3.4 Combining the reductions

We combine the reductions from Lemma 3.6, Lemma 3.10 and Lemma 3.11 to give the following reduction.

Lemma 3.12 (Main reduction)**.** *For all integer n, s, w, for all $\varepsilon \geq 1/\operatorname{poly}(w)$, there exists an explicit*

$$\left(O\left(s + \frac{\log n \log(nw)}{\log\log w - \log\log n} + \log w(\log\log\log w - \log\log \frac{\log w}{\log n/\varepsilon})\right), (8n)^{2\sqrt{\frac{\log w}{\log n}}}, \varepsilon\right)$$

-weighted pseudorandom reduction from $\mathcal{B}_{(n,s,w)}$ to $\mathcal{B}_{(O(1),O(\log w),w)}$.

Proof. We set up the reductions as the following, which is also illustrated in Figure 1.

We have the following reductions:

Let $(\mathsf{R}^{(1)}, w^{(1)})$ be a $(d_1, W_1, \varepsilon/4)$-weighted pseudorandom reduction from $\mathcal{B}_{(n,s,w)}$ to $\mathcal{B}_{(n_1,s_1,w)}$ given by Lemma 3.11, where

- $d_1 = O(\log w)$,
- $n_1 = \log^{1/2} w$,
- $s_1 = O(s + \frac{\log n \log(nw)}{\log\log w - \log\log n})$,
- $W_1 = (8n)^{\sqrt{\frac{\log w}{\log n}}+1}$.

Let $(\mathsf{R}^{(2)}, w^{(2)})$ be a $(s_2, W_2, \varepsilon/(4W_1))$-weighted pseudorandom reduction from $\mathcal{B}_{(n_1,s_1,w)}$ to $\mathcal{B}_{(n_2,s_2,w)}$ given by Lemma 3.6, where

- $d_2 = O(s + \frac{\log n \log(nw)}{\log\log w - \log\log n})$,
- $n_2 = \log^{1/2} w$,
- $s_2 \leq 2C_{GUV} \log(1/\varepsilon)$,
- $W_2 = 1$.

Using Lemma 2.2, $(\mathsf{R}^{(1)} \circ \mathsf{R}^{(2)}, w^{(1)} \cdot w^{(2)})$ is a $(d_1 + d_2, W_1, \varepsilon/2)$ weighted pseudorandom reduction from $\mathcal{B}_{(n,s,w)}$ to $\mathcal{B}_{(\log^{1/2} w, 2C_{GUV}\log(1/\varepsilon), w)}$. We set $\varepsilon' = (\varepsilon/W_1)^2$ and continue the reduction with the following parameters:

$n_3 = \log^{1/2} w, n_{i+1} = n_i^{1/3}$ for $i = 3, \ldots, l-1$. $n_l = \frac{\log w}{\log 1/\varepsilon'} = \frac{\log w}{\log 1/\varepsilon + \sqrt{\log n \log w}}$.

It is clear that

$$l = \log\log\log w - \log\log \frac{\log w}{\log n/\varepsilon} + O(1).$$

For $i = 3, \ldots, l-1$, let $(\mathsf{R}^{(i)}, w^{(i)})$ be a (d_i, K_i, ε')-weighted pseudorandom reduction from $\mathcal{B}_{(n_i, 2C_{GUV}\log(1/\varepsilon'), w)}$ to $\mathcal{B}_{(n_{i+1}, C_{large}\log w, w)}$ given by Lemma 3.10, setting constants $c = 3$, $C = \frac{2C_{GUV}\log(1/\varepsilon')}{\log w}$, $a = \log_w(1/\varepsilon')$ in Lemma 3.10, where

- $d_i = \log K_i = O(n_i^{1/2}\log n_i)$,
- $K_i = \exp(O(n_i^{1/2}\log n_i))$.

Notice that since $n_3 = \log^{1/2} w$ and n_i is monotonously decreasing, the condition $n_i \le \log^3 w$ is always satisfied, so the condition in Lemma 3.10 is always met.

For $i = 3, \ldots, l-2$, let $(\widehat{\mathsf{R}}^{(i)}, \widehat{w}^{(i)})$ be a $(\widehat{d}_i, \widehat{K}_i, \varepsilon')$-weighted pseudorandom reduction from $\mathcal{B}_{(n_{i+1}, C_{large}\log w, w)}$ to $\mathcal{B}_{(n_{i+1}, 2C_{GUV}\log(1/\varepsilon'), w)}$ given by Lemma 3.6, where

- $\widehat{d}_i = O(\log w)$,
- $\widehat{K}_i = 1$.

We invoke Lemma 2.3 to compose the $2l-7$ reductions, which gives $(\mathsf{R}^{(3)} \circ \widehat{\mathsf{R}}^{(3)} \circ \mathsf{R}^{(4)} \circ \widehat{\mathsf{R}}^{(4)} \circ \ldots \circ \mathsf{R}^{(l-1)}, w^{(3)} \cdot \widehat{w}^{(3)} \cdot w^{(4)} \cdot \widehat{w}^{(4)} \cdot \cdots \cdot w^{(l-1)})$. Also by Lemma 2.3, we know the reduction is a $(d^*, W^*, \varepsilon^*)$-weighted pseudorandom reduction from $\mathcal{B}_{(n,s,w)}$ to $\mathcal{B}_{(n_l, 2C_{GUV}\log(1/\varepsilon'), w)}$. The latter class is contained in $\mathcal{B}_{(O(1), C_{GUV}\log w, w)}$.The parameters $(d^*, W^*, \varepsilon^*)$ is shown as following:

- $d^* = \sum_{i=3}^{l-1} d_i = O(l\log w)$,
- $W^* = \prod_{i=3}^{l-1} W_i \cdot \widehat{W}_i = \exp(\sum_{i=3}^{l-1} n_i^{1/2}\log n_i) \le \exp(l\log^{1/3} w) < \exp(\log^{1/2} w)$,
- $\varepsilon^* = \sum_{i=3}^{l-1} \varepsilon' \cdot \prod_{j=3}^{i-1} W_j \cdot \widehat{W}_j \le l \cdot \varepsilon \cdot W < \varepsilon' \cdot \exp(\log^{1/2} w) < \varepsilon/(2W_1)$.

We invoke Lemma 2.2 again and compose the $(d_1+d_2, W_1, \varepsilon/2)$-weighted pseudorandom reduction from $\mathcal{B}_{(n,s,w)}$ to $\mathcal{B}_{(\log^{1/3} w, 2C_{GUV}\log(1/\varepsilon), w)}$ with the $(d^*, W^*, \varepsilon^*)$-weighted pseudorandom reduction from $\mathcal{B}_{(\log^{1/3} w, 2C_{GUV}\log(1/\varepsilon), w)}$ to $\mathcal{B}_{(O(1), C_{GUV}\log w, w)}$. We get the desired reduction with the following parameters:

- Seed length: $s_1 + s_2 + s^* = O(s + \frac{\log n \log(nw)}{\log\log w - \log\log n} + \log w(\log\log\log w - \log\log\frac{\log w}{\log n/\varepsilon})$,
- Weight: $W_1 \cdot W^* \le (8n)^{2\sqrt{\frac{\log w}{\log n}}}$,
- Error: $\varepsilon/2 + W_1 \cdot \varepsilon/(2W_1) = \varepsilon$.

The lemma follows. □

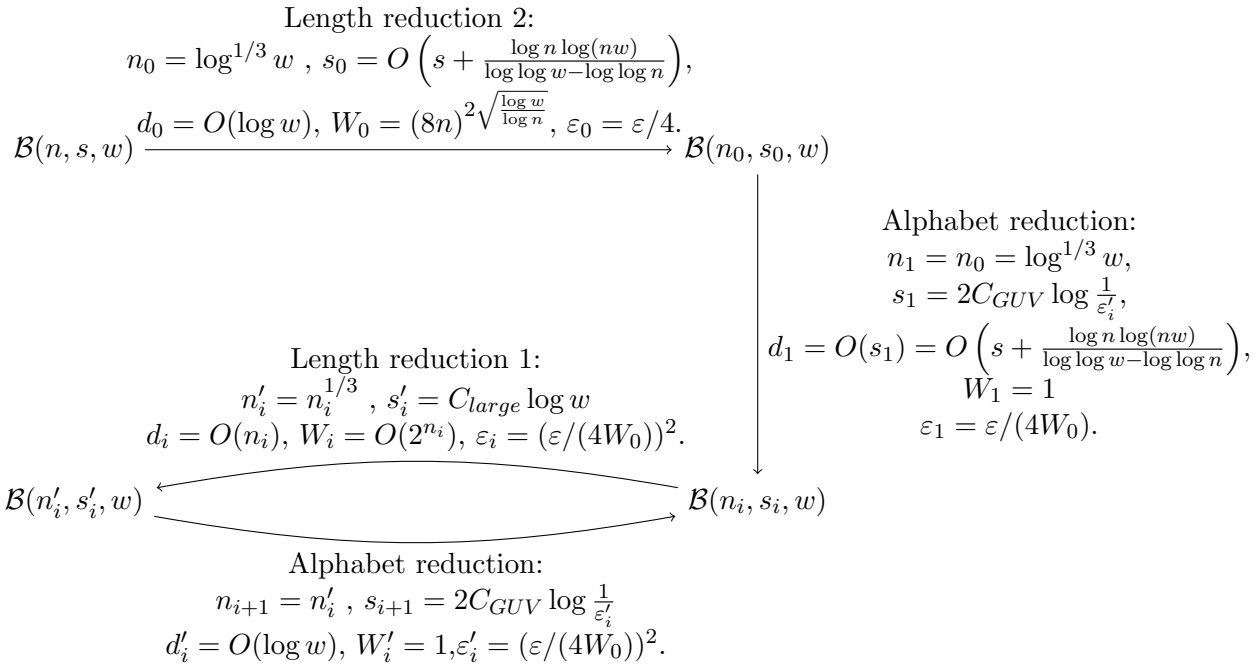

Figure 1: The recursion we use to iteratively reduce the ROBP in the construction of the WPRG Lemma 3.1. The arrows represent the reduction from an ROBP to simpler ROBPs.

The reduction naturally gives a WPRG, which is the PRG in Lemma 3.1

Lemma 3.13 (Lemma 3.1 restated)**.** *For all $C > 0, \varepsilon > 1/\operatorname{poly}(w)$, there exists an explicit ε-WPRG for $\mathcal{B}_{(n, C\log w, w)}$ with seed length $O\left(\frac{\log n \log(nw)}{\log\log w - \log\log n} + \log w(\log\log\log w - \log\log \frac{\log w}{\log n/\varepsilon})\right)$ and weight $(8n)^{2\sqrt{\frac{\log w}{\log n}}}$.*

Proof. Let (R, w) be the reduction from Lemma 3.12. We construct the WPRG (G, w) with the same weight function w and G is defined as $G(x, i) = \mathsf{R}(x, i)$. By Lemma 3.12, to fool the original ROBP we only need to provide randomness for the seed of (R, w) and also provide a random string x which can fool an arbitrary ROBP in $\mathcal{B}_{(O(1), C_{GUV}\log w, w)}$. We use true randomness for x and this only takes $O(\log w)$ random bits. Also the randomness used by (R, w) is $O\left(\frac{\log n \log(nw)}{\log\log w - \log\log n} + \log w(\log\log\log w - \log\log \frac{\log w}{\log n/\varepsilon})\right)$. So the total seed length is as stated. The weight also directly follows from Lemma 3.12. □

3.5 Reducing error beyond $1/\operatorname{poly}(w)$

To further reduce the error from $1/\operatorname{poly}(w)$ an arbitrary small ε, we use the sampler trick from [Hoz21]. The original technique applies to a PRG and gets a WPRG with a smaller error. Here we also use the technique but we apply it on a WPRG and get a WPRG with a smaller error, for the convenience of our analysis.

We shall first introduce the definition of an averaging sampler:

Definition 3.14 (Averaging Sampler)**.** *A (α, γ)-averaging sampler is a function* $\mathrm{Samp} : \{0,1\}^r \times \{0,1\}^p \to \{0,1\}^q$ *such that for every function $f : \{0,1\}^q \to [-1,1]$, it holds that:*

$$\Pr_{x \in \{0,1\}^r}\left[\left|2^{-p}\sum_{y \in \{0,1\}^p} f(\mathrm{Samp}(x,y)) - \mathbb{E}[f]\right| \geq \alpha\right] \leq \gamma.$$

Lemma 3.15 ([CL20] Appendix B, [Gol11, RVW00])**.** *For all $\alpha > 0, \gamma > 0$, there exists an explicit (α, γ)-averaging sampler with seed length $r = q + O(\log(1/\alpha) + \log(1/\gamma))$ and $p = O(\log(1/\alpha) + \log\log(1/\gamma))$.*

We now introduce the construction of a WPRG for class $\mathcal{B}_{(n,s,w)}$ with error ε. First, let (G_0, w_0) be a WPRG for $\mathcal{B}_{(n,s,w)}$ with error $1/(2w \cdot (n+1)^2)$ and weight W, which is constructed from Lemma 3.12. We then set the parameters as follows:

- $k = \frac{\log(n/\varepsilon)}{\log(nw)}$,
- $\alpha = 1/(W \cdot w^2 \cdot (n+1)^2)$,
- $\gamma = \varepsilon/(2(2n)^k \cdot (W)^{k+1} \cdot w^2)$.

We assume that Samp is a (α, γ)-averaging sampler, and let $K, n_{i,j}, w_{i,j}$ be as in Lemma 3.8. Our WPRG is constructed as follows:

$$\begin{cases} G(x, y_1, \ldots, y_k, i) = G_0(\mathrm{Samp}(x, y_1))_{n_{i,1}}, G_0(\mathrm{Samp}(x, y_2))_{n_{i,2}-n_{i,1}}, \ldots, G_0(\mathrm{Samp}(x, y_k))_{n_{i,k}-n_{i,k-1}} \\ w(x, y_1, \ldots, y_k, i) = \sigma_i K \cdot \prod_{j=1}^{k} w_0(\mathrm{Samp}(x, y_j)) \end{cases}$$

Now we show that the construction serves as a good WPRG for $\mathcal{B}_{(n,s,w)}$ with error ε.

Lemma 3.16. *For all n, s, w, assume there exists a W-bounded $1/(2w \cdot (n+1)^2)$-WPRG (G_0, w_0) for $\mathcal{B}_{(n,s,w)}$ with seed length d, and a (α, γ)-averaging sampler* $\mathrm{Samp} : \{0,1\}^r \times \{0,1\}^p \to \{0,1\}^d$ *with α, γ as defined above. Then there exists a ε-WPRG for $\mathcal{B}_{(n,s,w)}$ with seed length $r + kp$ and weight $W^{\log(n/\varepsilon)/\log(nw)} \cdot \mathrm{poly}(nw/\varepsilon)$.*

The proof of Lemma 3.16 is deferred to Appendix C.

Combining the lemma with Lemma 3.1, we prove the main theorem of the section:

Theorem 3.17 (Theorem 3.2 restated)**.** *For all integer n, s, w, there exists an explicit construction of a ε-WPRG for $\mathcal{B}_{(n,s,w)}$ with seed length*

$$O\left(s + \frac{\log n \log(nw)}{\log\log w - \log\log n} + \log w(\log\log\log w - \log\log\frac{\log w}{\log n/\varepsilon}) + \log(1/\varepsilon)\right)$$

and weight $\mathrm{poly}(nw/\varepsilon)$.

Proof. Let (G_0, w_0) be a $1/(2w \cdot (n+1)^2)$-WPRG for $\mathcal{B}_{(n,s,w)}$ with seed length

$$d = O\left(s + \frac{\log n \log(nw)}{\log\log w - \log\log n} + \log w(\log\log\log w - \log\log\frac{\log w}{\log n/\varepsilon})\right)$$

and weight $W = \mathrm{poly}(nw/\varepsilon)$ given by Lemma 3.12.

Let Samp be a $(1/(2W \cdot w^2 \cdot (n+1)^2), \varepsilon/(2(2n)^k \cdot W^k))$-averaging sampler with seed length $r = d + O(\log(nw/\varepsilon))$ and $p = O(\log(nw) + \log\log(1/\varepsilon))$ given by Lemma 3.15.

By Lemma 3.16, we get a ε-WPRG for $\mathcal{B}_{(n,s,w)}$ with seed length

$$r + (2k+1)p = O\left(s + \frac{\log n \log(nw)}{\log\log w - \log\log n} + \log w(\log\log\log w - \log\log\frac{\log w}{\log n/\varepsilon}) + \log(1/\varepsilon)\right)$$

and weight $\mathrm{poly}(nw/\varepsilon)$.

□

4 WPRG for Permutation Read-once Branching Programs

In this section we focus on WPRGs against permutation ROBPs. First we recall the definition of permutation ROBPs with only one accept node.

Definition 4.1 (Permutation ROBPs)**.** *A ROBP $f \in \mathcal{B}_{(n,s,w)}$ is a permutation ROBP if for every $i \in [n]$, $x \in \{0,1\}^s$, the matrix $f^{[i-1,i]}(x)$ is a permutation matrix.*

We denote the class of permutation ROBPs with unbounded width and only one accept node by $\mathcal{P}_{(n,s)}$, where n is the length s is the log-size of the alphabet.

Next we show our WPRG for $\mathcal{P}_{(n,s)}$.

Theorem 4.2. *There exists an ε-WPRG for $\mathcal{P}_{(n,s)}$ with seed length $s+O(\log n(\log\log n+\sqrt{\log(1/\varepsilon)})+\log(1/\varepsilon))$ and weight $2^{O(\log n\sqrt{\log(1/\varepsilon)})}$.*

Before we prove the theorem, we introduce some more definitions and lemmas.

Definition 4.3 (PSD norm)**.** *Let A be a $w \times w$ positive semi-definite matrix. The PSD norm on $\mathbb{R}^w$ with respect to A is defined as $\|x\|_A = \sqrt{x^T A x}$.*

Definition 4.4 (sv-approximation [APP+23])**.** *Let $\widetilde{W}$ and W be two $w \times w$ doubly stochastic matrices. We say that $\widetilde{W}$ is a ε-singular-value approximation of W, denoted by $\widetilde{W} \overset{sv}{\approx}_\varepsilon W$, if for all $x, y \in \mathbb{R}^w$,*

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

Definition 4.5 (fooling with sv-error)**.** *Let $(G, w) : \{0,1\}^s \to \{0,1\}^n \times \mathbb{R}$ be a WPRG. Let $\mathcal{C}$ be a family of matrix valued functions $\mathbf{B} : \{0,1\}^s \to \mathbb{R}^{w\times w}$. We say that G fools $\mathcal{C}$ with ε-sv-error if for all $\mathbf{B} \in \mathcal{C}$, $\sum_{z\in\{0,1\}^s}[\frac{1}{2^s}\mathbf{B}(G(z)) \cdot w(x)] \overset{sv}{\approx}_\varepsilon \mathbb{E}_{x\in\{0,1\}^n}[\mathbf{B}(x)]$.*

We describe a remarkable previous PRG against permutation ROBPs, given by [HPV21b]. From now on we may slightly abuse the notation $\mathcal{P}_{(n,s)}$ to let elements of it also denote matrix-valued functions.

Lemma 4.6 (Lemma 4.1 in [CHL+23], originally by [HPV21b])**.** *For all s, n, ε there exists a PRG (actually the INW generator) that fools $\mathcal{P}_{(n,s)}$ with ε-sv-error. The seed length is*

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

Remark. *We note that Lemma 4.1 in [CHL+23] only proves the case for $s = 2$, but their proof can naturally be generalized to handle arbitrary s. See Appendix E.*

Next we describe the key ingredient in previous error reductions for WPRGs against permutation ROBPs.

Lemma 4.7 (Claim 9.3 in [CHL+23]). *Let $n \in \mathbb{N}$, let $\tau \in \left(0, \frac{1}{64 \log^2 n}\right)$, and let $G : \{0,1\}^d \to (\{0,1\}^s)^n$ be a PRG that fools $\mathcal{P}_{(n,s)}$ with sv-error τ. Let $k \in \mathbb{N}$, and let*

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

Then there exists a WPRG $G^{(k)}, w^{(k)}$ that can be written in the form of

$$G^{(k)}(i, x_1, \ldots, x_r) = G(x_1)_{n_{i,1}}, G(x_2)_{n_{i,2}}, \ldots, G(x_r)_{n_{i,r}},$$
$$w^{(k)}(i) = \sigma(i) \cdot K.$$

where $K = n^{O(k)}, i \in [K], r = O(k \log n)$, $\sigma(i) \in \{-1, 0, 1\}$ and $0 \le n_{i,j} \le n$, such that $G^{(k)}$ fools $\mathcal{P}_{(n,s)}$ with entrywise error α.

4.1 One level of the reduction

We show that the previous lemma already gives a reduction:

Lemma 4.8. *For any integer n, k, s and any $\varepsilon > 0$, there exists $\tau = \Omega(\varepsilon^{\frac{2}{k+1}} / \log^2 n)$ such that there exists a $(O(k \log n), n^{O(k)}, \varepsilon)$-weighted pseudorandom reduction (R, w) from $\mathcal{P}_{(n,s)}$ to $\mathcal{P}_{(O(k \log n), s + O(\log n(\log\log n + \log 1/\tau)))}$.*

Proof. Let G be the INW PRG that fools $\mathcal{P}_{(n,s)}$ with τ-sv-error, such that $\tau = \min\{\frac{\varepsilon^{2/(k+1)}}{16 \log^2 n}, \frac{1}{64 \log^2 n}\}$. Then the seed length is $d = s + O(\log n(\log\log n + \log 1/\tau))$ by Lemma 4.6.

We invoke Lemma 4.7 with G and k, which gives $(G^{(k)}, w^{(k)})$. This WPRG fools $\mathcal{P}_{(n,s)}$ with entrywise error $(4 \cdot \sqrt{\tau} \cdot \log n)^{k+1} < \varepsilon$. We define the reduction $(\mathsf{R}, w) : \mathsf{R}_i(x_1, ..., x_r) = G^{(k)}(i, x_1, ..., x_r), w_i = w^{(k)}(i)$.

By Lemma 4.7, the weight of this reduction is bounded by $K = n^{O(k)}$. The seed length of the reduction is $\log K = O(k \log n)$, since we need $\log K = O(k \log n)$ bits to choose i.

Given any $f \in \mathcal{P}_{(n,s)}$, notice that each $f \circ \mathsf{R}_i$ is a ROBP of length $r = O(k \log n)$ and alphabet size $2^d = 2^{s + O(\log n(\log\log n + \log 1/\tau))}$. The reason is that the transition matrix of $f \circ \mathsf{R}_i$ from the $j-1$-th layer to the j-th layer on input $x \in \{0,1\}^d$ can be expressed as $(f \circ \mathsf{R}_i)^{[j-1,j]}(x)$. As $(f \circ \mathsf{R})^{[j-1,j]}(x) = f^{[n_{i,j-1}, n_{i,j}]}(G(x)_{n_{i,j} - n_{i,j-1}})$ and the latter is a product of $n_{i,j} - n_{i,j-1}$ permutation matrices, the transition matrix of $f \circ \mathsf{R}_i$ is also a permutation matrix. Hence $f \circ \mathsf{R}_i$ is a permutation ROBP. Also, note that each $f \circ \mathsf{R}_i$ only has one accept node since its accept node is the same as that of f.

Therefore, the reduction is a $(O(k \log n), O(n^k), \varepsilon)$-weighted pseudorandom reduction. □

Setting $k = \sqrt{\log(1/\varepsilon)}$ in Lemma 4.8, we immediately have the following lemma:

Lemma 4.9. *For any integer n, s and any $\varepsilon > 0$, there exists a $(O(\log n \sqrt{\log(1/\varepsilon)}), 2^{O(\log n \sqrt{\log(1/\varepsilon)})}, \varepsilon)$-weighted pseudorandom reduction from $\mathcal{P}_{(n,s)}$ to $\mathcal{P}_{(O(\log n \sqrt{\log(1/\varepsilon)}), s + O(\log n(\sqrt{\log(1/\varepsilon)} + \log\log n)))}$.*

Setting $k = O(\log^2 n)$ in Lemma 4.8, we immediately have the following lemma:

Lemma 4.10. *For any integer n, s and $\varepsilon > 0$, there exists a $(O(\log^3 n), 2^{O(\log^3 n)}, \varepsilon)$-weighted pseudorandom reduction from $\mathcal{P}_{(n,s)}$ to $\mathcal{P}_{(\log^3 n, s + O(\frac{\log(1/\varepsilon)}{\log n} + \log n \log\log n))}$.*

4.2 Recursion of reductions

We give a multi-level recursive procedure which gradually reduces the program from $\mathcal{P}_{(n,s)}$ to $\mathcal{P}_{(O(1),s+O(\log n(\log\log n+\sqrt{\log(1/\varepsilon)})))}$ with error ε, for any n, s, ε.

We start by reducing the length of the permutation ROBP to $O(\log n\sqrt{\log(1/\varepsilon)})$ and the weight to $2^{O(\log n\sqrt{\log(1/\varepsilon)})}$ with error $\varepsilon/2$. Previous works use an INW PRG to fool the reduced ROBP, which leads to an extra double logarithmic factor in the seed length. Here we use the recursion instead to avoid this extra factor.

Lemma 4.11 (Reduction for permutation ROBP)**.** *For any integer n, s and any $\varepsilon > 0$, there exists a $\left(O(\log n\sqrt{\log(1/\varepsilon)}), 2^{\log n\sqrt{\log(1/\varepsilon)}}, O(\varepsilon)\right)$-weighted pseudorandom reduction from $\mathcal{P}_{(n,s)}$ to $\mathcal{P}_{(O(1),s+O(\log n(\log\log n+\sqrt{\log(1/\varepsilon)})+\log(1/\varepsilon)))}$.*

Proof. Let $(\mathsf{R}^{(0)}, w^{(0)}$ be the $(d_0, W_0, \varepsilon/2)$-weighted pseudorandom reduction from $\mathcal{P}_{(n,s)}$ to $\mathcal{P}_{(n_1,s_1)}$ such that $n_1 = O(\log n\sqrt{\log(1/\varepsilon)})$ and $s_1 = s + O(\log n(\log\log n + \sqrt{\log(1/\varepsilon)}))$. The reduction is guaranteed by Lemma 4.9 and have the following parameters:

1. $d_0 = O(\log n\sqrt{\log(1/\varepsilon)})$,
2. $W_0 = 2^{O(\log n\sqrt{\log(1/\varepsilon)})}$.

Now we define a new parameter $\varepsilon' = \varepsilon/2W_0$. Let

$$n_0 = n, n_i = \log^3 n_{i-1}$$

for each $i = 1, \ldots, l$ and $n_l = 32768$. It is clear that $l \le \log\log n$.

For each $i = 1, \ldots, l$, let $(\mathsf{R}^{(i)}, w^{(i)})$ be the $(d_i, W_i, (\varepsilon')^2)$-weighted pseudorandom reduction from $\mathcal{P}_{(n_i,s_i)}$ to $\mathcal{P}_{(n_{i+1},s_{i+1})}$, which is guaranteed by Lemma 4.10. The reduction has the following parameters:

1. $d_i = O(\log^3 n_i)$,
2. $W_i = 2^{\log^3 n_i}$,
3. $s_{i+1} = s_i + O(\frac{\log(1/\varepsilon')}{\log n_i} + \log n_i \log\log n_i) = s_i + O(\frac{\log(1/\varepsilon)}{\log n_i})$.

Using Lemma 2.3, we first composite $(\mathsf{R}^{(1)}, w^{(1)}), \ldots, (\mathsf{R}^{(l)}, w^{(l)})$ to get a $(d^*, W^*, \varepsilon^*)$-weighted pseudorandom reduction $(\mathsf{R}^{(*)}, w^{(*)})$ from $\mathcal{P}_{(n_0,s_0)}$ to $\mathcal{P}_{(n_l,s_l)}$ with $\mathsf{R}^{(*)} = \mathsf{R}^{(1)} \circ \ldots \circ \mathsf{R}^{(l)}$ and $w^{(*)} = w^{(1)} \cdot \ldots \cdot w^{(l)}$. We have the following parameters:

1. $d^* = d_1 + \ldots + d_l \le O(l \log^3 n_1) \le O(\log^{0.1}(n/\varepsilon))$,
2. $W^* = W_1 \cdot \ldots \cdot W_l \le 2^{O(l \cdot \log^3 n_1)} \le 2^{O(\log^{0.1}(n/\varepsilon))}$,
3. $\varepsilon^* = \sum_{i=1}^{l} (\varepsilon')^2 \cdot \prod_{j=0}^{i-1} W_j \le (\varepsilon')^2 \cdot W^* \cdot l \le \varepsilon'/2$,
4. $s_l = s_0 + \sum_{i=1}^{l} (s_i - s_{i-1}) = s_0 + \sum_{i=1}^{l} O(\frac{\log(1/\varepsilon')}{\log n_i}) = s_0 + O(\log 1/\varepsilon')$.

The last item holds because n_i decreases rapidly. Because $n_{i-1} = 2^{\sqrt[3]{n_i}}$, when $n_i \geq 2^{15} = 32768$, We have $n_{i-1} \geq n_i^2$. When $n_l \geq 32768$, $\sum_{i=1}^{l} \frac{\log(1/\varepsilon')}{\log n_i} \leq \log 1/\varepsilon' \cdot (1/\log n_l + 1/(2\log n_l) + 1/(4\log n_l) + \cdots + 1/(2^l \log n_l)) \leq \log 1/\varepsilon'$.

Finally, we composite $(\mathsf{R}^{(*)}, w^{(*)})$ with $(\mathsf{R}^{(0)}, w^{(0)})$ to get the final reduction (R, w) from $\mathcal{P}_{(n,s)}$ to $\mathcal{P}_{(O(1),s_l))}$. By Lemma 2.2, the parameters are:

1. $d = d^* + d_0 = O(\log n\sqrt{\log(1/\varepsilon)})$,
2. $W = W^* \cdot W_0 \leq 2^{O(\log n\sqrt{\log(1/\varepsilon)})}$,
3. $\varepsilon = \varepsilon/2 + \varepsilon'/2 \cdot W_0 \leq \varepsilon$,
4. $s = s_l = s_0 + O(\log 1/\varepsilon') = s + O(\log n(\log\log n + \sqrt{\log(1/\varepsilon)}) + \log(1/\varepsilon))$.

□

One can also see our reduction strategy for the above lemma through Figure 4.2.

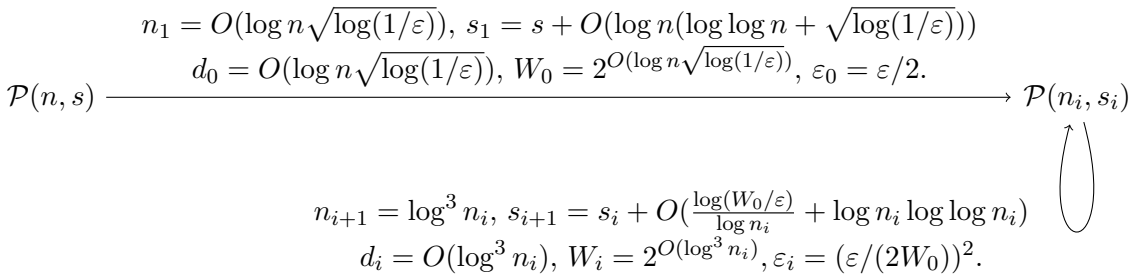

Figure 2: Reduction for permutation ROBPs in Lemma 4.11.

Given this lemma, we immediately have the main theorem of this section:

Theorem 4.12 (Theorem 4.2 restated)**.** *There exists an ε-WPRG for $\mathcal{P}_{(n,s)}$ with seed length $s + O(\log n(\log\log n + \sqrt{\log(1/\varepsilon)}) + \log(1/\varepsilon))$ and weight $2^{O(\log n\sqrt{\log(1/\varepsilon)})}$.*

Proof. Let (R, w) be the reduction from Lemma 4.11. We construct the WPRG (G, w) with the same weight function w, and G is defined as $G(x, i) = \mathsf{R}(x, i)$. By Lemma 4.11, to fool the original ROBP we only need to provide randomness for the seed of (R, w) and also provide a random string x which can fool an arbitrary ROBP in $\mathcal{P}_{(O(1),s+O(\log n(\log\log n+\sqrt{\log(1/\varepsilon)})+\log(1/\varepsilon)))}$. We can use another ε error INW generator from [HPV21a] to generate this constant length x and this only takes $s + O\left(\log n(\log\log n + \sqrt{\log(1/\varepsilon)}) + \log(1/\varepsilon)\right)$ random bits. Also the randomness used by (R, w) is $O(\log n\sqrt{\log(1/\varepsilon)})$. So the total seed length is as stated. The weight also directly follows from Lemma 4.11.

□

It is well known that a WPRG fools a Permutation ROBP with an arbitrary number of accept nodes and error ε if it fools the program with only one accept node with error ε/w. Therefore, we have the following corollary:

Theorem 4.13. *There exists an ε-WPRG for length n, width w, alphabet size (out degree) 2^s Permutation ROBPs having an arbitrary number of accept nodes, with seed length $s + O(\log n(\log\log n + \sqrt{\log(w/\varepsilon)}) + \log(w/\varepsilon))$ and weight $2^{O(\log n\sqrt{\log(w/\varepsilon)})}$.*

5 Derandomizing Regular Branching Programs

In this section, we give our improved derandomization for short-wide regular ROBPs. When $n \leq 2^{\log^{1/2} w}$ and $\varepsilon = 1/\operatorname{poly}(w)$, our result provides an $O(\log \frac{w}{\varepsilon})$ space derandomization, which is optimal up to a constant factor. We achieve this by constructing a new derandomization algorithm that runs in space $O(\log(w/\varepsilon) + \log n \sqrt{\log(w/\epsilon)})$, adapted from our WPRG for permutation ROBPs from the Section 4.

We remark that for the special case of binary regular ROBPs, one can transform them to permutation ROBPs in logspace, see Appendix D, and then run the WPRG of the previous section. But for the general case, i.e. arbitrary alphabet size, the problem turns out to be more complex. In the rest of this section, we mainly focus on the general case.

5.1 More preliminaries

We recall some standard definitions in the literature of [RVW00, RV05, AKM+20, CHL+23, CL24].

Definition 5.1 (Regular bigraph)**.** *A bigraph is a triple $G = (U, V, E)$, where U and V are two sets of vertices and $E \subseteq U \times V$ is a set of edges going from U to V. A bigraph is called d-regular if every vertex in U has d outgoing edges and every vertex in V has d incoming edges.*

The transition matrix of a regular bigraph is a the matrix $\mathbf{M} \in \mathbb{R}^{V \times U}$ such that $\mathbf{M}_{v,u}$ is the fraction of edges going from u that go to v.

Definition 5.2 (One-way labeling)**.** *A one-way labeling of a d-regular bigraph G assigns a label $i \in [d]$ to each edge in G such that for every vertex $u \in U$, the labels of the outgoing edges of u are distinct. If G has a one way labeling, we say that $G[u, i] = v$ if the outgoing edge of u that is labeled i goes to v.*

Definition 5.3 (Two-way labeling)**.** *A two-way labeling of a d-regular bigraph G is a labeling of the edges of G such that:*

- *Every edge (u, v) has two labels in $[d]$, the 'outgoing label' and the 'incoming label'.*
- *For every vertex $u \in U$, the outgoing labels of the outgoing edges of u are distinct.*
- *For every vertex $v \in V$, the incoming labels of the incoming edges of v are distinct.*

Definition 5.4 (Rotation map)**.** *Let $G = (U, V, E)$ be a d-regular bigraph with a two-way labeling. The rotation map of G is a function $\mathsf{Rot}_G : U \times [d] \to V \times [d]$ such that $\mathsf{Rot}_G(u, i) = (v, j)$ if there is an edge $(u, v) \in E$ with the outgoing label i and the incoming label j.*

Definition 5.5 (Derandomized Product)**.** *Let $G_1 = (U, V, E_1)$ and $G_2 = (V, T, E_2)$ be two 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 one way labeling. The derandomized product of G_1, G_2 and H is a $(c \cdot d)$-regular bigraph with one-way labeling denoted by $G_1 \textcircled{\scriptsize p}_H G_2$ defined as follows. To compute $(G_1 \textcircled{\scriptsize p}_H G_2)[v_0, (i_0, j_0)]$ for $v_0 \in U$ and $(i_0, j_0) \in [d] \times [c]$:*

- *Let $(v_1, i_1) = \mathsf{Rot}_{G_1}(v_0, i_0)$.*
- *Let $i_2 = H[i_1, j_0]$.*
- *Let $v_2 = G_2[v_1, i_2]$.*
- *Output $(G_1 \textcircled{\scriptsize p}_H G_2)[v_0, (i_0, j_0)] =: v_2$.*

Definition 5.6 (Two-way labeling of derandomized product)**.** *Let $G_1 = (U, V, E_1)$ and $G_2 = (V, T, E_2)$ be two d-regular bigraphs where both G_1 and G_2 have two-way labeling. Let $H = ([d], [d], E_H)$ be a c-regular bigraph with two-way labeling. We define the two-way labeling of the derandomized product $G_1 ⓟ_H G_2$ as follows. To compute the rotation map $\mathsf{Rot}_{G_1 ⓟ_H G_2}$ for any vertex $v_0 \in U$ and any pair $(i_0, j_0) \in [d] \times [c]$, we do the following:*

- *Let $(v_1, i_1) = \mathsf{Rot}_{G_1}(v_0, i_0)$.*
- *Let $(i_2, j_1) = \mathsf{Rot}_H(i_1, j_0)$.*
- *Let $(v_2, i_3) = \mathsf{Rot}_{G_2}(v_1, i_2)$.*
- *Output $\mathsf{Rot}_{G_1 ⓟ_H G_2}(v_0, (i_0, j_0)) = (v_2, (i_3, j_1))$.*

To ensure that the derandomized product behaves like the direct concatenation of the two bigraphs, we need H to be a spectral expander.

Definition 5.7 (spectral expander)**.** *Let $H = (U, V, E_H)$ be a d-regular bigraph with transition matrix $W_H \in \mathbb{R}^{U \times V}$, where $|U| = |V|$. Define $J \in \mathbb{R}^{U \times V}$ where $J_{i,j} = 1/|U|$ for all $i \in U, j \in V$. Then the spectral expansion of H is denoted by $\lambda(H)$ and defined as follows:*

$$\lambda(H) = \|W_H - J\|_2$$

We also need the definitions for regular branching programs:

Definition 5.8 (Regular Branching Program)**.** *Let f be a ROBP in $\mathcal{B}_{n,s,w}$. We call f a regular branching program if all vertices in the graph of f except the vertices in the first layer have precises 2^s incoming edges.*

A general regular branching program may not have a two-way labeling. If we equip it with a two-way labeling for each step of the transition, we call it a regular ROBP with two-way labeling.

Definition 5.9 (Regular ROBP with Two-way labeling)**.** *A regular ROBP with two-way labeling is a pair (f, ℓ_{in}) where f is a regular ROBP in $\mathcal{B}_{n,s,w}$ with vertex set $V = V_0 \cup V_1 \cup \cdots \cup V_n$, edge set $E = E_1 \cup \cdots \cup E_n$ and edge labeling $\ell_{out} : E \to [2^s]$. The function $\ell_{in} : E \to [2^s]$ map every edge to another label. Furthermore, for every vertex $i \in [n]$, l_{in} and l_{out} restricted on E_i is a two-way labeling of the bigraph $G_i := (V_{i-1}, V_i, E_i)$.*

We neglect ℓ_{out} in the notation since we can always consider ℓ_{out} to be the canonical order given by the ROBP.

We denote the class of all regular ROBPs with two-way labeling $\mathcal{R}^{tw}_{n,s,w}$.

We recall the derandomization algorithm for regular branching programs in [CHL+23].

Let (f, ℓ_{in}) be a regular ROBP with two-way labeling in $\mathcal{R}^{tw}_{n,s,w}$. For each $i \in [\log n]$, let $H_t = ([2^s \cdot c^{t-1}], [2^s \cdot c^t], E_{H_t})$ be a c-regular bigraph with $\lambda(H_t) \le \lambda$. Define $\widetilde{G}_{j \to j+1} := (V_{t-1}, V_t, E_t)$, which is a 2^s-regular bigraph with two-way labeling. Define $E(SC_n) = \{(j, j + 2^t) : j \in [n - 2^t], t \in [\log n], 2^t \text{ is the largest power of 2 dividing } j\}$. For each $(j, j + 2^t) \in E(SC_n)$ recursively define the graph $\widetilde{G}_{j \to j+2^t}$ as follows:

$$\widetilde{G}_{j \to j+2^t} := \widetilde{G}_{j \to j+2^{t-1}} ⓟ_{H_i} \widetilde{G}_{j+2^{t-1} \to j+2^t}$$

The following lemma shows that the transition matrix of the final graph $\widetilde{G}_{0 \to n}$ is close to the product of the transition matrices of the intermediate graphs.

Lemma 5.10 (Lemma A.20 of [CHL+23]). *Let $n, d, c \in \mathbb{N}$. For each $t \in \{0, 1, \ldots, \log n\}$ and each $j \in [n - 2^t]$, let $\widetilde{G}_{j+2^{t-j}} = \left(V_j, V_{j+2^t}, E_{j+2^{t-j}}\right)$ be a $(d \cdot c^t)$-regular bigraph with a two-way labeling. Furthermore, let $\lambda \in \left(0, \frac{1}{6 \log^2 n}\right)$, and for each $t \in [\log n]$, let $H_t = \left([d \cdot c^{t-1}], [d \cdot c^{t-1}], E_{H_t}\right)$ be a c-regular bigraph with a one-way labeling satisfying $\lambda(H_t) \leq \lambda$. Assume also that for each $t \in [\log n]$ and each $j \in [n - 2^t]$, we have*

$$\widetilde{G}_{j \to j+2^t} = \widetilde{G}_{j \to j+2^{t-1}} \textcircled{p}_{H_t} \widetilde{G}_{j+2^{t-1} \to j+2^t}, \forall t \geq 1,$$

and

$$\widetilde{G}_{j \to j+1} = G_{j \to j+1},$$

(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 $\widetilde{G}_{i \to j}$, and let

$$\mathbf{W}_{j \leftarrow i} = \widetilde{\mathbf{W}}_{j \leftarrow j-1} \cdots \widetilde{\mathbf{W}}_{i+1 \leftarrow i}$$

Then

$$\widetilde{\mathbf{W}}_{n \leftarrow 0} \overset{sv}{\approx}_{11\lambda \cdot \log n} \mathbf{W}_{n \leftarrow 0}.$$

Theorem 5.11 (Claim A.23 of [CHL+23]). *The following algorithm computes*

$$\mathsf{Rot}_{\widetilde{G}_{0 \to n}}(v_0, (x, e_1, \ldots, e_{\log n}))$$

1. *For $i = 1$ to n:*
   (a) *Update $(v, x) \leftarrow \mathsf{Rot}_{\widetilde{G}_{i-1 \to i}}(v, x)$, 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 $(x, e_1, \ldots, e_{t-1}) \leftarrow \mathsf{Rot}_{H_t}((x, e_1, \ldots, e_{t-1}), e_t)$.*
2. *Output (v, e).*

We also need the efficiently computable spectral expander H_t. The following lemma shows that such a spectral expander exists. The construction of H_t is based on Margulis-Gabber-Galil graphs [Mar73, GG81].

Lemma 5.12 (Space-efficient expanders, Lemma A.22 of [CHL+23]). *For every $d \in \mathbb{N}$ that is a power of two, for every $\lambda \in (0, 1)$, there is a bigraph $H = \left([d], [d], E_H\right)$ with a two-way labeling satisfying the following:*

- $\lambda(H) \leq \lambda$.
- *H is c-regular where c is a power of two and $c \leq \mathrm{poly}(1/\lambda)$.*
- Rot_H *can be evaluated in space that is linear in its input length, i.e., space $O(\log(d/\lambda))$.*

We use the following error reduction polynomial in our algorithm, which gives a good sv-approximation for a sequence of transitions.

Theorem 5.13 (Recursion [CL24]). *Let $\{A_i\}_{i=1}^n \subset \mathbb{R}^{w\times w}$ be a sequence of doubly stochastic matrices. Let $\{B_{i,j}\}_{i,j=0}^n \subset \mathbb{R}^{w\times w}$ be a family of matrices such that $B_{i,j} \overset{sv}{\approx}_{\varepsilon/(10\log n)} A_{i+1}\dots A_j$ Assuming that $B_{i-1,i} = A_i$ for all i. Then for any $k \in \mathbb{N}$, there exists $K = O((2n)^k)$, $t = O(k\log n)$, a set of indices $n_{i,j}$ for each $i \in [K], j \in [t]$ with $0 \le n_{i,1} \le \dots \le n_{i,t} = n$ and a set of signs $\sigma_i \in \{-1,0,1\}$ for each $i \in [K]$ such that:*

$$\sum_{i\in[K]} \sigma_i \cdot B_{0,n_{i,1}} B_{n_{i,1},n_{i,2}} \dots B_{n_{i,t-1},n_{i,t}} \overset{sv}{\approx}_{\varepsilon^k} A_1 A_2 \dots A_n.$$

5.2 The algorithm

We give our derandomization algorithm for regular branching programs.

Theorem 5.14. *There exists an algorithm that takes as input a regular ROBP and a parameter $\varepsilon > 0$, and outputs an approximation of its acceptance probability with error ε. Furthermore, if the regular ROBP has length n, width w and alphabet size 2^s, then the algorithm runs in space $O(s + \log n(\log\log n + \sqrt{\log(w/\varepsilon)}) + \log(w/\varepsilon))$.*

Proof. To define the algorithm, we need to prepare three things in advance: the two-way labeling, the error reduction, and the rotation algorithm.

First, the input of the algorithm is a regular ROBP f without two-way labeling. However, we can easily convert it to a regular ROBP G with two-way labeling by assigning an arbitrary label to each incoming edge. (i.e., $\ell_{in}(e) = i$ if e is the i-th incoming edge of the vertex.). This can be done in space $O(s + \log(nw))$.

Second, we need to predetermine the sets of indices and signs used in a recursive error reduction procedure. We use parameters similar to those in Section 4. We set $n_0 = n, n_1 = O(\log n\sqrt{\log(w/\varepsilon)})$. For $i = 2, 3, \dots$, let $n_i = \lceil \log^3 n_{i-1} \rceil$. We take l to be the first integer such that $n_l \le 2^{15}$. Notice that $l = o(\log\log n)$.

Now for each $i \in [l]$, we prepare a group of parameters for invoking Theorem 5.13. We use n_i, n_{i+1} as the n, r respectively in Theorem 5.13. When we apply it to n_0, n_1, we set $\alpha_0 = \varepsilon/2w$ as the target error and $k_0 = \sqrt{\log(1/\alpha_0)}$, $\tau_0 = \min\{\frac{\varepsilon^{2/(k_0+1)}}{16\log^2 n_0}, \frac{1}{64\log^2 n_0}\}$ as the original error. Theorem 5.13 gives a number $W_0 \in \mathbb{N}$,a set of indices $\{n_{j,t}^{(0)}\}_{j\in[W_0],t\in[n_1]}$ and a set of signs $\{\sigma_j\}_{j\in[W_0]}$. After that, we set $\varepsilon' = \varepsilon^2/(2wW_0)$. Then for every $i \in [l-1]$, we set $\alpha_i = \varepsilon'$ and $k_i = \log^2 n_i$, $\tau_i = \min\{\frac{(\varepsilon')^{2/(k_i+1)}}{16\log^2 n_i}, \frac{1}{64\log^2 n_i}\}$. Theorem 5.13 gives a number $W_i \in \mathbb{N}$, a set of indices $\{n_{j,t}^{(i)}\}_{j\in[W_i],t\in[n_{i+1}]}$ and a set of signs $\{\sigma_j\}_{j\in[W_i]}$.

We stress that one can assume each $\{n_{j,t}^{(i)}\}_{j\in[W_i],t\in[n_{i+1}]}$ and $\{\sigma_j\}_{j\in[W_i]}$ satisfies the following properties:

For each $i = 0, 1, 2, \dots, l-1$, for any doubly stochastic matrices $A_1, \dots, A_{n_i}$ and $B_{j,q}$ such that $B_{p,q} \overset{sv}{\approx}_{\tau_i} A_{p+1}\dots A_q$ and $B_{p-1,p} = A_p$ for all $p \in [n_i]$. Then let

$$A' = \sum_{j\in[W_i]} \sigma_j \cdot B_{0,n_{j,1}^{(i)}} B_{n_{j,1}^{(i)},n_{j,2}^{(i)}} \dots B_{n_{j,n_{i+1}-1}^{(i)},n_{j,n_{i+1}}^{(i)}}.$$

It holds that A' approximates $A_1 \dots A_{n_i}$ with entry-wise error at most α_i, by Theorem 5.13.

The third part of the preparation is an implementation of the rotation algorithm Theorem 5.11. We use the algorithm for every level i, to get those approximation matrices with sv-error τ_i.

For any $i = 0, 1, 2, \dots, l-1$, to use Lemma 5.10 for the correctness of Theorem 5.11, we need to prepare a group of parameters n_i, λ_i, d_i, c_i. Let λ_i be $\tau_i/(11\log n_i)$. For each $t \in [\log n_i]$, we

use $H_t^{(i)}$ as the t-th λ_i-spectral expander used in the lemma, where every $H_t^{(i)}$ can be achieved by Lemma 5.12. The parameter c_i is set to be the degree of $H_t^{(i)}$, which is the same for all t. We let $d_0 = 2^s$. Let $d_{i+1} = d_i + \lceil \log n_i \rceil \cdot \lceil \log c_i \rceil$ for all $i = 0, 1, \ldots, l-1$. One can see later that the parameter d_i is the degree of the branching program after the first i levels of reduction.

Now we have prepared $l, \{n_i\}_{i=0}^{l}, \{W_i\}_{i=0}^{l}, \{n_{j,t}^{(i)}\}_{i\in[l],j\in[W_i],t\in[n_{i+1}]}, \{\sigma_j\}_{j\in[n_i]}, \{d_i\}_{i=0}^{l}, \{c_i\}_{i=1}^{l}$ and graphs $\{H_t^{(i)}\}_{i\in[l],t\in[\log n_i]}$. Notice that all these parameters can be computed in logspace, so we will directly use them in the algorithm description. Then we construct our algorithm that approximates the acceptance probability of the regular ROBP. In general the algorithm adapts the WPRG of the previous section and runs the ROBP on the output of the WPRG but with the following two adjustments:

- every time when the WPRG needs to conduct a one-step walk on an expander, then the algorithm instead conducts a corresponding rotation on that expander.
- every time when running one step of the ROBP using an output symbol of the WPRG, the algorithm conducts a corresponding rotation on the ROBP.

For completeness of the description, the whole algorithm is given as Algorithm 1.

In Algorithm 1, for every $j \in [n]$, $G_{j\leftarrow j-1}$ is the bipartite graph (V_{j-1}, V_j, E_j) of input ROBP f, as described in Definition 5.9. $\mathsf{Rot}_{H_p^{(i)}}$ and $\mathsf{Rot}_{G_{j\leftarrow j-1}}$ are defined by Definition 5.4. We use the $x^{(i)}$ to denote a temporary variable that stores a label in the following way. $x^{(l)} \in [d_l] = [d_0] \times [c_0]^{\log n_0} \times [c_1]^{\log n_1} \times \cdots \times [c_{l-1}]^{\log n_{l-1}}$. For every $i \in \{0, 1, ..., l-1\}$, $x^{(i)}$ denotes the prefix of $x^{(l)}$ in $[d_i] = [d_0] \times [c_0]^{\log n_0} \times \cdots \times [c_{i-1}]^{\log n_{i-1}}$, and $e_t^{(i)}$ is in $[c_i]$ for every $t \in [\log n_i]$ such that $x^{(i)} = (x^{(i-1)}, e_1^{(i-1)}, ..., e_{\log n_{i-1}}^{(i-1)})$.

Algorithm 1: Approximate the acceptance probability of a regular ROBP

Input: A regular ROBP f, a starting vertex v_0, an accepting vertex v_{end}, an error parameter $\varepsilon > 0$.

Output: An approximation of the probability that f reaches v_{end} from v_0.

Set a counter $ans \leftarrow 0$.

foreach $j_0 \in [W_0], j_1 \in [W_1], \ldots, j_{l-1} \in [W_{l-1}]$(Enumerate all possible summands in the error reduction polynomial) **do**

  foreach $(x_1^{(l)}, x_2^{(l)}, \ldots, x_{n_l}^{(l)})$ (Enumerate seeds), where each $x_t^{(l)} \in [d_l] = [d_0] \times [c_0]^{\log n_0} \times [c_1]^{\log n_1} \times \cdots \times [c_{l-1}]^{\log n_{l-1}}$ **do**

    Let $x^{(l)}$ be a temporary variable and $x^{(l)} \leftarrow x_1^{(l)}$.

    Initiate position pointers for each level $t_0 \leftarrow 0, t_1 \leftarrow 0, \ldots, t_l \leftarrow 0$.

    Let $v \leftarrow v_0$ be a temporary variable recording the current vertex.

    while $t_l < n_l$ **do**

      $t_0 \leftarrow t_0 + 1$.

      $(v, x^{(0)}) \leftarrow \mathsf{Rot}_{G_{t_0 \leftarrow t_0 - 1}}(v, x^{(0)})$.

      Update pointers: For $i = 1$ to l, if $t_{i-1} = n^{(i-1)}_{j_{i-1}, t_i + 1}$, we let $t_i \leftarrow t_i + 1$. Let i be the largest i such that t_i is updated.

      if $i < l$ **then**

        Let $p \in [\log n_i]$ be the smallest positive integer such that $t_i - n^{(i)}_{j_i, t_{i+1}}$ is not a multiple of 2^p.

        Update $(x^{(i)}, e_1^{(i)}, e_2^{(i)}, \ldots, e_{p-1}^{(i)}) \leftarrow \mathsf{Rot}_{H_p^{(i)}}((x^{(i)}, e_1^{(i)}, e_2^{(i)}, \ldots, e_{p-1}^{(i)}), e_p^{(i)})$.

      else if $i = l$ **then**

        $x^{(l)} \leftarrow x^{(l)}_{t_l + 1}$

    if $v = v_{end}$ **then**

      Add $\sigma_{j_0} \cdot \sigma_{j_1} \cdots \sigma_{j_{l-1}}$ to ans.

Normalize ans by $ans \leftarrow ans / d_l$, return ans.

The following figure gives an example of how our algorithm 1 computes the desired rotation maps.

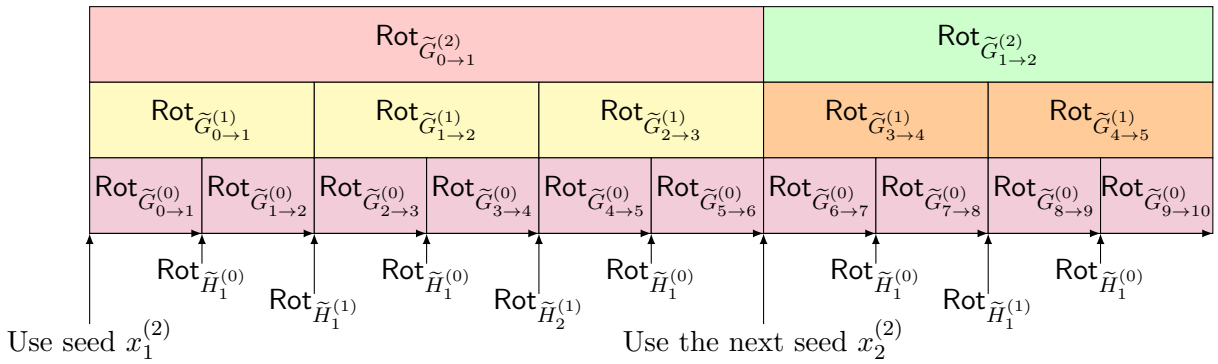

Figure 3: The figure shows our recursion process to compute $\mathsf{Rot}_{\widetilde{G}^{(l)}_{1\to2}}\left(\mathsf{Rot}_{\widetilde{G}^{(l)}_{0\to1}}\left(v_0, x_1^{(l)}\right), x_2^{(l)}\right)$ when $l = 2, n_l = 2$. The indices j_0, j_1, j_2 are omitted.

Next we prove the correctness of the algorithm.

Let $\widetilde{G}^{(0)}_{t-1\to t}$ denote the bipartite graph with two-way labeling of the input from the $(t-1)$-th layer to the t-th layer. We recursively define $\widetilde{G}^{(i)}_{j_0,j_1,\ldots,j_{i-1},t-1\to t}$ as the graph obtained by applying Theorem 5.11 to $\widetilde{G}^{(i-1)}_{j_0,j_1,\ldots,j_{i-2},a\to a+1},\ldots,\widetilde{G}^{(i-1)}_{j_0,j_1,\ldots,j_{i-2},b-1\to b}$ where $b=n^{(i)}_{j_{i-1},t}, a=n^{(i)}_{j_{i-1},t-1}$ with the aforementioned parameters n_i,λ_i,d_i,c_i. Let $\widetilde{\mathbf{W}}^{(i)}_{j_0,j_1,\ldots,j_{i-1},t\leftarrow t-1}$ be the transition matrix of $\widetilde{G}^{(i)}_{j_0,j_1,\ldots,j_{i-1},t-1\to t}$.

The output of Algorithm 1 is a weighted sum of $\widetilde{\mathbf{W}}^{(l)}_{j_0,j_1,\ldots,j_{i-1},n_l\leftarrow n_l-1}\cdots\widetilde{\mathbf{W}}^{(l)}_{j_0,j_1,\ldots,j_{i-1},1\leftarrow 0}$, as shown in the following equation:

$$
\begin{aligned}
&ans\\
&=\sum_{\substack{j_0\in[W_0],j_1\in[W_1],\\ \ldots,j_{l-1}\in[W_{l-1}]}}\left(\prod_{i=0}^{l-1}\sigma_{j_i}\right)\cdot\mathbb{E}_{x_1^{(l)},x_2^{(l)},\ldots,x_{n_l}^{(l)}\in[d_l]}\\
&\quad\left[\mathbf{1}\left[\mathsf{Rot}_{\widetilde{G}^{(l)}_{j_0,j_1,\ldots,j_{l-1},n_l-1\to n_l}}\left(\cdots\mathsf{Rot}_{\widetilde{G}^{(l)}_{j_0,j_1,\cdots,j_{l-1},1\to 2}}\left(\mathsf{Rot}_{\widetilde{G}^{(l)}_{j_0,j_1,\cdots,j_{l-1},0\to 1}}\left(v_0,x_1^{(l)}\right),x_2^{(l)}\right)\ldots,x_{n_l}^{(l)}\right)=v_{end}\right]\right]\\
&=\sum_{\substack{j_0\in[W_0],j_1\in[W_1],\\ \ldots,j_{l-1}\in[W_{l-1}]}}\left(\prod_{i=0}^{l-1}\sigma_{j_i}\right)\cdot\left[\widetilde{\mathbf{W}}^{(l)}_{j_0,j_1,\ldots,j_{i-1},n_l\leftarrow n_l-1}\cdots\widetilde{\mathbf{W}}^{(l)}_{j_0,j_1,\ldots,j_{i-1},1\leftarrow 0}\right]_{v_0,v_{end}}.
\end{aligned}
$$

By Lemma 5.10, for any $i\in[l], j_0,j_1,\ldots,j_{i-1}\in[W_0],[W_1],\ldots,[W_{i-1}]$, letting $b=n^{(i)}_{j_{i-1},t}, a=n^{(i)}_{j_{i-1},t-1}$, we have the following approximation.

$$
\widetilde{\mathbf{W}}^{(i)}_{j_0,j_1,\ldots,j_{i-1},t\leftarrow t-1}\overset{sv}{\approx}_{\tau_{i-1}}\widetilde{\mathbf{W}}^{(i-1)}_{j_0,j_1,\ldots,j_{i-2},b\leftarrow b-1}\widetilde{\mathbf{W}}^{(i-1)}_{j_0,j_1,\ldots,j_{i-2},b-1\leftarrow b-2}\cdots\widetilde{\mathbf{W}}^{(i-1)}_{j_0,j_1,\ldots,j_{i-2},a\leftarrow a+1}.
$$

These matrices satisfy the condition of Theorem 5.13. By the construction of the $\{n^{(i)}_{j,t}\}$ and $\{\sigma_j\}$, we have that for each $i\in 0,1,\ldots,l-1$ and each $j_0,j_1,\ldots,j_{i-1}$,

$$
\left|\left[\widetilde{\mathbf{W}}^{(i)}_{\substack{j_0,\ldots,j_{i-1}\\ n_i\leftarrow n_i-1}}\cdots\widetilde{\mathbf{W}}^{(i)}_{\substack{j_0,\ldots,j_{i-1}\\ 1\leftarrow 0}}\right]_{v_0,v_{end}}-\sum_{j_i\in[W_i]}\left[\sigma_{j_i}\cdot\widetilde{\mathbf{W}}^{(i+1)}_{\substack{j_0,\ldots,j_i\\ n_{i+1}\leftarrow n_{i+1}-1}}\cdots\widetilde{\mathbf{W}}^{(i+1)}_{\substack{j_0,\ldots,j_i\\ 1\leftarrow 0}}\right]_{v_0,v_{end}}\right|\le\alpha_i.
$$

By summing over all the errors, we have the following:

$$
\begin{aligned}
&\left| ans - \left[\widetilde{\mathbf{W}}^{(0)}_{n_0 \leftarrow n_0 - 1} \cdots \widetilde{\mathbf{W}}^{(0)}_{1 \leftarrow 0} \right]_{v_0, v_{end}} \right| \\
=&\left| \sum_{i=0}^{l-1} \sum_{\substack{j_0 \in [W_0], j_1 \in [W_1], \\ \ldots, j_{i-1} \in [W_{i-1}]}} \left(\prod_{k=0}^{i-1} \sigma_{j_k} \right) \right. \\
&\left. \cdot \left(\left[\widetilde{\mathbf{W}}^{(i)}_{\substack{j_0, \ldots, j_{i-1} \\ n_i \leftarrow n_i - 1}} \cdots \widetilde{\mathbf{W}}^{(i)}_{\substack{j_0, \ldots, j_{i-1} \\ 1 \leftarrow 0}} \right]_{v_0, v_{end}} - \sum_{j_i \in [W_i]} \left[\sigma_{j_i} \cdot \widetilde{\mathbf{W}}^{(i+1)}_{\substack{j_0, \ldots, j_i \\ n_{i+1} \leftarrow n_{i+1} - 1}} \cdots \widetilde{\mathbf{W}}^{(i+1)}_{\substack{j_0, \ldots, j_i \\ 1 \leftarrow 0}} \right]_{v_0, v_{end}} \right) \right| \\
\leq &\sum_{i=0}^{l-1} \sum_{\substack{j_0 \in [W_0], j_1 \in [W_1], \\ \ldots, j_{i-1} \in [W_{i-1}]}} \alpha_i \\
\leq &\alpha_0 + W_0 \cdot \alpha_1 + \ldots + W_0 \cdot W_1 \cdots W_{l-1} \cdot \alpha_l.
\end{aligned}
$$

By our parameters, this entry-wise error is at most ε/w. Therefore, the total error is at most ε when the accept node set is an arbitrary subset of $V_n = [w]$.

Then we prove the space complexity of the algorithm. Similar to the proof of Lemma 4.11, $d_l = O(\log n(\log\log n + \sqrt{\log(w/\varepsilon)}) + \log(w/\varepsilon))$, which is exactly the seed length in that proof. The algorithm shall store ans, j_i, n_i for each $i \in [l-1]$ and $x^{(l)}$, which costs no more than $O(s + \log n(\log\log n + \sqrt{\log(w/\varepsilon)}) + \log(w/\varepsilon))$ space. Calculating a two-way labeling and computing $\mathsf{Rot}_{G^{(0)}_{t \leftarrow t-1}}$ cost $O(s + \log(nw))$ space. The computation of $\mathsf{Rot}_{H^{(i)}_t}$ costs $O(|x^{(i)}|)$ space according to Lemma 5.12. Therefore, the total space complexity is $O(s + \log n(\log\log n + \sqrt{\log(w/\varepsilon)}) + \log(w/\varepsilon))$. □

References

[AKM+20] AmirMahdi Ahmadinejad, Jonathan Kelner, Jack Murtagh, John Peebles, Aaron Sidford, and Salil Vadhan. High-precision Estimation of Random Walks in Small Space. In 2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS), pages 1295–1306. IEEE, 2020.

[APP+23] AmirMahdi Ahmadinejad, John Peebles, Edward Pyne, Aaron Sidford, and Salil Vadhan. Singular Value Approximation and Sparsifying Random Walks on Directed Graphs. In 2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS), pages 846–854. IEEE, 2023.

[Arm98] Roy Armoni. On the Derandomization of Space-Bounded Computations. In Michael Luby, José D. P. Rolim, and Maria Serna, editors, Randomization and Approximation Techniques in Computer Science, pages 47–59, Berlin, Heidelberg, 1998. Springer.

[BCG19] 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, 2019.

[BHPP22] 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.

[BRRY14] Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. Pseudorandom Generators for Regular Branching Programs. SIAM Journal on Computing, 43(3):973–986, January 2014.

[BV10] Joshua Brody and Elad Verbin. The coin problem and pseudorandomness for branching programs. In 2010 IEEE 51st Annual Symposium on Foundations of Computer Science, pages 30–39. IEEE, 2010.

[CDR+21] 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, page 22. Schloss Dagstuhl-Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 2021.

[CDST23] Gil Cohen, Dean Doron, Ori Sberlo, and Amnon Ta-Shma. Approximating Iterated Multiplication of Stochastic Matrices in Small Space. In Proceedings of the 55th Annual ACM Symposium on Theory of Computing, pages 35–45, Orlando FL USA, June 2023. ACM.

[CH22] Kuan Cheng and William M. Hoza. Hitting sets give two-sided derandomization of small space. Theory of Computing, 18(1):1–32, 2022.

[CHHL19] Eshan Chattopadhyay, Pooya Hatami, Kaave Hosseini, and Shachar Lovett. Pseudorandom generators from polarizing random walks. Theory of Computing, 15(1):1–26, 2019.

[CHL+23] Lijie Chen, William M. Hoza, Xin Lyu, Avishay Tal, and Hongxun Wu. Weighted Pseudorandom Generators via Inverse Analysis of Random Walks and Shortcutting. In 2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS), pages 1224–1239. IEEE, 2023.

[CL20] Eshan Chattopadhyay and Jyun-Jie Liao. Optimal Error Pseudodistributions for Read-Once Branching Programs. In 35th Computational Complexity Conference (CCC 2020). Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2020.

[CL24] Eshan Chattopadhyay and Jyun-Jie Liao. Recursive Error Reduction for Regular Branching Programs. In 15th Innovations in Theoretical Computer Science Conference (ITCS 2024). Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2024.

[CW24] Kuan Cheng and Yichuan Wang. BPL $\subseteq$ L-AC^1. In 39th Computational Complexity Conference (CCC 2024). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024.

[De11] Anindya De. Pseudorandomness for permutation and regular branching programs. In 2011 IEEE 26th Annual Conference on Computational Complexity, pages 221–231. IEEE, 2011.

[GG81] Ofer Gabber and Zvi Galil. Explicit constructions of linear-sized superconcentrators. Journal of Computer and System Sciences, 22(3):407–420, 1981.

[Gol11] Oded Goldreich. A sample of samplers: A computational perspective on sampling. In Studies in complexity and cryptography: miscellanea on the interplay between randomness and computatio, pages 302–332. Springer, 2011.

[GUV09] Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. Unbalanced expanders and randomness extractors from Parvaresh–Vardy codes. Journal of the ACM, 56(4):1–34, June 2009.

[Hoz21] 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.

[HPV21a] William M. Hoza, Edward Pyne, and Salil Vadhan. Pseudorandom Generators for Unbounded-Width Permutation Branching Programs. In 12th Innovations in Theoretical Computer Science Conference (ITCS 2021). Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2021.

[HPV21b] William M. Hoza, Edward Pyne, and Salil Vadhan. Pseudorandom generators for unbounded-width permutation branching programs. In 12th Innovations in Theoretical Computer Science Conference (ITCS 2021). Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2021.

[HZ20] William M. Hoza and David Zuckerman. Simple optimal hitting sets for small-success rl. SIAM Journal on Computing, 49(4):811–820, 2020.

[INW94] Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing - STOC '94, pages 356–364, Montreal, Quebec, Canada, 1994. ACM Press.

[KNP11] Michal Kouckỳ, Prajakta Nimbhorkar, and Pavel Pudlák. Pseudorandom generators for group products. In Proceedings of the forty-third annual ACM symposium on Theory of computing, pages 263–272, 2011.

[KNW08] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. Revisiting Norm Estimation in Data Streams. arXiv preprint arXiv:0811.3648, 2008.

[LPV23] 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). Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2023.

[Mar73] Grigorii Aleksandrovich Margulis. Explicit constructions of concentrators. Problemy Peredachi Informatsii, 9(4):71–80, 1973.

[Nis92a] Noam Nisan. Pseudorandom generators for space-bounded computation. Combinatorica, 12(4):449–461, December 1992.

[Nis92b] Noam Nisan. RL $\subseteq$ SC. In Proceedings of the twenty-fourth annual ACM symposium on Theory of computing, pages 619–623, 1992.

[NZ96] Noam Nisan and David Zuckerman. Randomness is Linear in Space. Journal of Computer and System Sciences, 52(1):43–52, February 1996.

[PP23] Aaron (Louie) Putterman and Edward Pyne. Near-Optimal Derandomization of Medium-Width Branching Programs. In Proceedings of the 55th Annual ACM Symposium on Theory of Computing, pages 23–34, 2023.

[PRZ23] Edward Pyne, Ran Raz, and Wei Zhan. Certified Hardness vs. Randomness for Log-Space. In 2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS), pages 989–1007. IEEE, 2023.

[PV21] Edward Pyne and Salil Vadhan. Pseudodistributions That Beat All Pseudorandom Generators (Extended Abstract). In 36th Computational Complexity Conference (CCC 2021). Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2021.

[RSV13] Omer Reingold, Thomas Steinke, and Salil Vadhan. Pseudorandomness for regular branching programs via fourier analysis. In International Workshop on Approximation Algorithms for Combinatorial Optimization, pages 655–670. Springer, 2013.

[RV05] Eyal Rozenman and Salil Vadhan. Derandomized Squaring of Graphs. In International Workshop on Approximation Algorithms for Combinatorial Optimization, pages 436–447. Springer, 2005.

[RVW00] Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In Proceedings 41st Annual Symposium on Foundations of Computer Science, pages 3–13. IEEE, 2000.

[Ste12] Thomas Steinke. Pseudorandomness for permutation branching programs without the group theory. In Electronic Colloquium on Computational Complexity (ECCC), volume 19, page 6, 2012.

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

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

A A proof for Lemma 3.5

We reprove the lemma for completeness.

Definition A.1 (total variation)**.** *Let A and B be two random variables defined on a common probability space X. The total variation distance between A and B is defined as*

$$d_{TV}(A,B) = \frac{1}{2}\sum_{x\in X} \left|\Pr[A=x] - \Pr[B=x]\right|.$$

Definition A.2 (Min-Entropy)**.** *Let X be a random variable. The min-entropy of X is defined as*

$$H_\infty(X) = -\log_2\left(\max_{x\in X}\Pr[X=x]\right).$$

Definition A.3 (Conditional Min-Entropy [Vad12] Problem 6.7)**.** *Let X and A be two random variables. The conditional min-entropy of X given Y is defined as*

$$\tilde{H}_\infty(X|Y) = -\log_2\left(\mathop{\mathbb{E}}_{y\in Y}\sup_{x\in X}\Pr[X=x|Y=y]\right).$$

We need the following lemmas.

Lemma A.4 (Chain rule of Conditional Min-Entropy [Vad12] Problem 6.7)**.** *If $|supp(A)| \leq 2^s$, then $\widetilde{H}_\infty(X|A) \geq H_\infty(X) - s$.*

Lemma A.5 (Problem 6.8 of [Vad12])**.** *Let $Ext : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ be a (k,ε)-extractor. If $\tilde{H}_\infty(X|A) \geq k$, then*

$$d_{TV}\left((Ext(X,U_d),A),(U_m,A)\right) \leq 3\varepsilon.$$

(Here U_m and U_d are independent uniform random variables of length m and d respectively.)

Lemma A.6 (Data Processing Inequality)**.** *Let X, Y be two random variables in the same probability space. Let A be a random variable that is independent of both X and Y. Let f be a function. Then*

$$d_{TV}(f(X,A),f(Y,A)) \leq d_{TV}(X,Y).$$

Now we can prove the lemma.

Lemma A.7 (Lemma 3.5 restated)**.** *Let $s \geq \log w$. Assume there exists a $(2s, \frac{\varepsilon}{3n})$-extractor $\textsc{Ext} : \{0,1\}^{3s} \times \{0,1\}^d \to \{0,1\}^s$. Let X and $Y_1, \ldots, Y_n$ be independent uniform random variables. Then the following construction*

$$\mathsf{NZ}(X,Y) = \textsc{Ext}(X,Y_1), \ldots, \textsc{Ext}(X,Y_n)$$

fools any $f \in \mathcal{B}_{(n,s,w)}$ with error at most ε.

Proof. Let $f \in \mathcal{B}_{(n,s,w)}$. Let X and $Y_1, \ldots, Y_n$ be independent uniform random variables. We use a hybrid argument. Define U_i to be independent uniform random variables of length s for each $i \in [n]$. Define $Z_i = \textsc{Ext}(X, Y_i)$ for each $i \in [n]$. Define R_i to be the random variable over V_i, which represents the distribution of the state of f on input $U_1, \ldots, U_i$. Define $\widetilde{R}_i$ to be the random variable over V_i which represents the distribution of the state of f on input $Z_1, \ldots, Z_i$. We will show that $d_{TV}(R_i, \widetilde{R}_i) \leq \frac{i\cdot\varepsilon}{n}$ for all $i \in [n]$.

The base case $i = 0$ is trivial, as both R_0 and $\widetilde{R}_0$ represent the initial state of f. Assume the statement holds for $i-1$. For the case i, notice that $|\text{supp}(\widetilde{R}_{i-1})| \leq w \leq 2^s$. By the chain rule Lemma A.4,

$$\tilde{H}_\infty(X|\widetilde{R}_{i-1}) \geq 2s.$$

Therefore Lemma A.5 implies

$$d_{TV}((Z_i, \widetilde{R}_{i-1}), (U_i, \widetilde{R}_{i-1})) \leq \frac{\varepsilon}{n}.$$

By the data processing inequality, we have

$$d_{TV}((U_i, \widetilde{R}_{i-1}), (U_i, R_{i-1})) \leq d_{TV}(\widetilde{R}_{i-1}, R_{i-1}) \leq \frac{(i-1)\varepsilon}{n}.$$

Using the triangle inequality, we have

$$d_{TV}((Z_i, \widetilde{R}_{i-1}), (U_i, R_{i-1})) \leq \frac{i\varepsilon}{n}.$$

Denote the transition function of f from V_{i-1} to V_i as T_i. Then $\widetilde{R}_i = T_i(Z_i, \widetilde{R}_{i-1})$ and $R_i = T_i(U_i, R_{i-1})$. Using the data processing inequality again, we have

$$d_{TV}(R_i, \widetilde{R}_i) \leq d_{TV}((U_i, R_{i-1}), (Z_i, \widetilde{R}_{i-1})) \leq \frac{i\varepsilon}{n}.$$

Therefore, $d_{TV}(R_n, \widetilde{R}_n) \leq \varepsilon$ and the lemma is proved. □

B A proof for Theorem 3.8

We prove Theorem 3.8 using the Richardson Iteration for completeness. The Richardson iteration is a method to obtain a finer approximation of L^{-1} from a coarse approximation B of L^{-1} and the invertible matrix L itself.

Lemma B.1. *Let $L \in \mathbb{R}^{m\times m}$ be an invertible matrix and $B \in \mathbb{R}^{m\times m}$ such that $\|B - L^{-1}\| \leq \varepsilon_0$ for a submultiplicative norm $\|\cdot\|$. For any non-negative integer k, define*

$$R(B, L, k) = \sum_{i=0}^{k} (I - BL)^i B$$

Then $\|L^{-1} - R(B, L, k)\| \leq \|L^{-1}\| \cdot \|L\|^{k+1} \cdot \varepsilon_0^{k+1}$.

Proof.

$$\begin{aligned}
\|L^{-1} - R(B, L, k)\| &= \left\| \left(I - \sum_{i=0}^{k} (I - BL)^i BL \right) L^{-1} \right\| \\
&= \left\| (I - BL)^{k+1} L^{-1} \right\| \\
&\leq \|I - BL\|^{k+1} \cdot \left\| L^{-1} \right\| \\
&\leq \left\| L^{-1} \right\| \cdot \|L\|^{k+1} \cdot \varepsilon_0^{k+1}
\end{aligned}$$

□

Theorem B.2 (Theorem 3.8 restated)**.** *Let $\{A_i\}_{i=1}^n \subset \mathbb{R}^{w\times w}$ be a sequence of matrices. Let $\{B_{i,j}\}_{i,j=0}^n \subset \mathbb{R}^{w\times w}$ be a family of matrices such that for every $i + 1 < j$, $\|B_{i,j} - A_{i+1} \dots A_j\| \leq \varepsilon/(n+1)^2$ for some submultiplicative norm $\|\cdot\|$, $\|A_i\| \leq 1$ for all i and also $B_{i-1,i} = A_i$ for all i. Then for any odd $k \in \mathbb{N}$, there exists a $K = (8n)^{k+1}$, a set of indices $\{n_{i,j}\}_{i\in[K],j\in[k]}$ with $0 \leq n_{i,1} \leq \dots \leq n_{i,k} = n$, and signs $\sigma_i \in \{-1, 0, 1\}, i \in [K]$ such that (We set $B_{i,i} = I$ for all i):*

$$\left\| A - \sum_{i\in[K]} \sigma_i \cdot B_{0,n_{i,1}} B_{n_{i,1},n_{i,2}} \dots B_{n_{i,k-1},n_{i,k}} \right\| \leq \varepsilon^{(k+1)/2} \cdot (n+1).$$

Proof. Define $L \in \mathbb{R}^{(n+1)w\times(n+1)w}$ and $B \in \mathbb{R}^{(n+1)w\times(n+1)w}$ as follows:

$$L = \begin{pmatrix} I & 0 & 0 & \dots & 0 & 0 \\ -A_1 & I & 0 & \dots & 0 & 0 \\ 0 & -A_2 & I & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -A_n & I \end{pmatrix}, B = \begin{pmatrix} I & 0 & 0 & \dots & 0 & 0 \\ B_{0,1} & I & 0 & \dots & 0 & 0 \\ B_{0,2} & B_{1,2} & I & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ B_{0,n} & B_{1,n} & B_{2,n} & \dots & B_{n-1,n} & I \end{pmatrix}$$

This means that

$$L^{-1} = \begin{pmatrix} I & 0 & 0 & \dots & 0 & 0 \\ A_1 & I & 0 & \dots & 0 & 0 \\ A_1A_2 & A_2 & I & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ A_1A_2 \dots A_n & A_2 \dots A_n & A_3 \dots A_n & \dots & A_n & I \end{pmatrix}$$

By assumption, there exists a submultiplicative norm $\|\cdot\|$ on $\mathbb{R}^{w\times w}$. We induce a submultiplicative norm on $\|\cdot\|_A$ on $\mathbb{R}^{(n+1)w\times(n+1)w}$: Let $M=(M_{i,j})_{i,j=1}^{n+1}\in\mathbb{R}^{(n+1)w\times(n+1)w}$, where each $M_{i,j}\in\mathbb{R}^{w\times w}$. Define $\|M\|_A=\max_{j\in[n+1]}\sum_{i=1}^{n+1}\|M_{i,j}\|$. It is easy to verify that $\|\cdot\|_A$ is a submultiplicative norm, and $\|M\|_A=\|M\|_1$ when $w=1$.

Beacuse $\|B_{i,j}-A_{i+1}\dots A_j\|\le\varepsilon/(n+1)^2$ for all $i+1<j$, we have $\|B-L^{-1}\|_A\le\varepsilon/(n+1)$. Also, Because $\|A_i\|\le 1$ for all i, we have $\|L\|_A\le n+1$ and $\|L^{-1}\|_A\le n+1$.

Define $M=R(B,L,(k-1)/2)$, then

$$\|L^{-1}-M\|_A\le\|L^{-1}\|_A\cdot\|L\|_A^{\frac{k+1}{2}}\cdot\left(\frac{\varepsilon}{n+1}\right)^{\frac{k+1}{2}}\le\varepsilon^{(k+1)/2}\cdot(n+1).$$

Expand $M=(M_{i,j})_{i,j=1}^{n+1}$, where each $M_{i,j}\in\mathbb{R}^{w\times w}$. We foucus on $M_{1,n+1}$, which is a $w\times w$ matrix and $\|M_{1,n+1}-A_1A_2\dots A_n\|\le\|L^{-1}-M\|_A\le\varepsilon^{(k+1)/2}\cdot(n+1)$.

To express $M_{1,n+1}$, we define

$$\Delta_{i,j}=\begin{cases}B_{i,j-1}A_j-B_{i,j} & i<j,\\ 0 & i\ge j.\end{cases}$$

Then we have

$$M_{1,n+1}=B_{0,n}+\sum_{j=1}^{(k-1)/2}\sum_{0<r_1<\dots<r_j<n}\Delta_{0,r_1}\Delta_{r_1,r_2}\cdots\Delta_{r_{j-1},r_j}B_{r_j,n}.$$

Defining $M_{i,j}^{(0)}=B_{i,j-1}A_j=B_{i,j-1}B_{j-1,j}$ and $M_{i,j}^{(1)}=B_{i,j}$, we have

$$M_{1,n+1}=B_{0,n}+\sum_{j=1}^{(k-1)/2}\sum_{0<r_1<\dots<r_j<n}\sum_{t_1,\dots,t_j\in\{0,1\}}(-1)^{t_1+\dots+t_j}M_{0,r_1}^{(t_1)}M_{r_1,r_2}^{(t_2)}\dots M_{r_j,n}^{(t_j)}.$$

Encoding $j,r_1,\dots,r_j,t_1,\dots,t_j$ into a single index $i\in[K]$, rewrite $M_{0,r_1}^{(t_1)}M_{r_1,r_2}^{(t_2)}\dots M_{r_j,n}^{(t_j)}$ as $B_{0,n_{i,1}}B_{n_{i,1},n_{i,2}}\dots B_{n_{i,k-1},n_{i,k}}$ for some $n_{i,1}\le\dots\le n_{i,k}=n$, define $\sigma_i=(-1)^{t_1+\dots+t_j}$, we have the desired result.

Finally, we bound K by $\frac{k-1}{2}\cdot(n+1)^{(k-1)/2}\cdot 2^{(k-1)/2}\le(8n)^{k+1}$.

□

C A Proof for Lemma 3.16

In this section we provide a proof for Lemma 3.16, which is similar to that of [Hoz21]. We start by sampling a random matrix using a sampler, using a union bound.

Lemma C.1. *Let* $\mathrm{Samp}:\{0,1\}^p\times\{0,1\}^r\to\{0,1\}^q$ *be a* (α,γ)*-averaging sampler. Then for every matrix valued function* $f:\{0,1\}^q\to\mathbb{R}^{w\times w}$ *such that* $\|f\|_1\le C$*, we have:*

$$\Pr_{x\in\{0,1\}^r}\left[\left\|2^{-p}\sum_{y\in\{0,1\}^p}f(\mathrm{Samp}(x,y))-\mathbb{E}[f]\right\|_1\ge C\alpha w\right]\le w^2\gamma.$$

Proof.

$$
\begin{aligned}
&\Pr_{x\in\{0,1\}^r}\left[\left\|2^{-p}\sum_{y\in\{0,1\}^p} f(\mathrm{Samp}(x,y)) - \mathbb{E}[f]\right\|_1 \geq C\alpha w\right] \\
=&\Pr_{x\in\{0,1\}^r}\exists_{i\in[w]}\left[\sum_{j\in[w]}\left[2^{-p}\sum_{y\in\{0,1\}^p} f(\mathrm{Samp}(x,y))_{i,j} - \mathbb{E}[f]_{i,j}\right] \geq C\alpha w\right] \\
\leq&\Pr_{x\in\{0,1\}^r}\exists_{i\in[w],j\in[w]}\left[\left|2^{-p}\sum_{y\in\{0,1\}^p} f(\mathrm{Samp}(x,y))_{i,j} - \mathbb{E}[f]_{i,j}\right| \geq C\alpha\right] \\
\leq&\sum_{i\in[w],j\in[w]}\Pr_{x\in\{0,1\}^r}\left[\left|C^{-1}2^{-p}\sum_{y\in\{0,1\}^p} f(\mathrm{Samp}(x,y))_{i,j} - C^{-1}\mathbb{E}[f]_{i,j}\right| \geq \alpha\right] \\
\leq& w^2\gamma.
\end{aligned}
$$

□

Lemma C.2 (Lemma 3.16 restated)**.** *For all n, s, w, assume there exists a W-bounded $1/(2w\cdot(n+1)^2)$-WPRG (G_0, w_0) for $\mathcal{B}_{(n,s,w)}$ with seed length d, and a (α,γ)-averaging sampler* $\mathrm{Samp}: \{0,1\}^r\times\{0,1\}^p\to\{0,1\}^d$ *with α,γ as defined above. Then there exists a ε-WPRG for $\mathcal{B}_{(n,s,w)}$ with seed length $r+kp$ and weight $W^{\log(n/\varepsilon)/\log(nw)}\cdot\mathrm{poly}(nw/\varepsilon)$.*

Proof. Take any $f\in\mathcal{B}_{(n,s,w)}$. Let $A_i = \mathbb{E}f^{[i-1,i]}(G_0(U))$ for $i\in[n]$. Define $B_{i,j} = \frac{1}{2^p}\sum_{z\in\{0,1\}^d} w_0(z)\cdot f^{[i,j]}(G(z))$. For any $x\in\{0,1\}^r$, we define $B^x_{i,j} = \frac{1}{2^p}\sum_{y\in\{0,1\}^p} w_0(\mathrm{Samp}(x,y))\cdot f^{[i,j]}((G(\mathrm{Samp}(x,y)))_{j-i})$.

We call a seed x to be 'good' iff for all $i,j\in[n]$, $\|B_{i,j}-B^x_{i,j}\|_1\leq 1/(2w\cdot(n+1))$. Otherwise, we call x to be 'bad'.

Since w_0 is W-bounded, $\|w_0(z)\cdot f^{[i,j]}(G(z))\|_1$ is at most W. Note that $W\alpha w\leq 1/(2w\cdot(n+1)^2)$. By Lemma C.1, for any $i,j\in[n]$,

$$\Pr_{x\in\{0,1\}^r}[\|B_{i,j}-B^x_{i,j}\|_1 > 1/(2w\cdot(n+1))]\leq w^2\gamma = \varepsilon/(2(2n)^k\cdot W^k).$$

By a union bound, the probability $\Pr_{x\in\{0,1\}^r}[x \text{ is bad}]$ is at most $\varepsilon/(2(2n)^k\cdot W^k)$.

For a good seed x, we have $\|A_i\dots A_j - B^x_{i,j}\|_1\leq\|A_i\dots A_j - B_{i,j}\|_1+\|B_{i,j}-B^x_{i,j}\|_1\leq 1/(w\cdot(n+1)^2)$. By Theorem 3.8, we have:

$$\left\|A_1\cdots A_n - \sum_{i=1}^{k}\sigma_i B^x_{0,i_1}B^x_{i_1,i_2}\dots B^x_{i_{k-1},i_k}\right\|_1\leq((n+1)w)^{-k}\cdot(n+1)\leq\varepsilon/2.$$

For a bad seed, we can upper-bound $\|B^x_{i,j}\|_1$ by K. Therefore, we have:

$$\left\|A_1\cdots A_n - \sum_{i=1}^{K}\sigma_i B^x_{0,i_1}B^x_{i_1,i_2}\dots B^x_{i_{k-1},i_k}\right\|_1\leq 1+K\cdot W^k$$

Combining the two cases, we have the following inequality:

$$\begin{aligned}
&\left|\mathbb{E}f - \frac{1}{K\cdot 2^{kp+r}}\sum_{x\in\{0,1\}^r}\sum_{y_1,\ldots,y_k\in\{0,1\}^p} w(x,y_1,\ldots,y_k,i)f(G(x,y_1,\ldots,y_k))\right| \\
\leq&\left\|A_1\cdots A_n - \frac{1}{2^r}\sum_{x\in\{0,1\}^r}\sum_{i=1}^{K}\sigma_i B^x_{0,i_1}B^x_{i_1,i_2}\ldots B^x_{i_{k-1},i_k}\right\|_1 \\
\leq&\frac{1}{2^r}\sum_{\substack{x\in\{0,1\}^r\\ x \text{ is bad}}}\left\|A_1\cdots A_n - \sum_{i=1}^{K}\sigma_i B^x_{0,i_1}B^x_{i_1,i_2}\ldots B^x_{i_{k-1},i_k}\right\|_1 \\
&+\frac{1}{2^r}\sum_{\substack{x\in\{0,1\}^r\\ x \text{ is good}}}\left\|A_1\cdots A_n - \sum_{i=1}^{K}\sigma_i B^x_{0,i_1}B^x_{i_1,i_2}\ldots B^x_{i_{k-1},i_k}\right\|_1 \\
\leq&\varepsilon/2 + (1+K\cdot W^k)\cdot\gamma \\
\leq&\varepsilon.
\end{aligned}$$

The lemma follows. □

D Transforming regular ROBPs to permutation ROBPs in the binary edge label setting

Lemma D.1. *There is an algorithm which on inputting a regular ROBP P with length n width w and binary edge labels, outputs a permutation ROBP P' with length n width w and binary edge labels. The algorithm has workspace $O(\log(nw))$.*

Proof. P' has the same vertex set as that of P. The algorithm transforms each layer i of P in the order $i = 0, 1, 2, \ldots, n$. For layer i, consider the subgraph S between V_i and V_{i+1}. The algorithm handles each node $v \in V_i$ in order. For each $v \in V_i$, if it is not reachable by any previous node of V_i in S, which can be done in logspace, then it does the following traverse in S. Starting from v, choose an edge with label 0, and then go to the corresponding neighbor. For each next neighbor u visited, use ℓ to record the label of the edge e it just traverses to this neighbor. If the other adjacent edge e' the neighbor has a label $\ell' = \ell$ then flip its label. Then traverse through this edge e' to go to the next neighbor. Go on doing this until it reaches v again.

Notice that starting from v we can visit every node of $V_i \cup V_{i+1}$ that is reachable from v in the above traverse. Because for each next neighbor u, it can only be either v or a node which is not visited before in this traverse, since except v, every node visited already has both adjacent edges visited, and the traverse only visits edge that is not visited before.

Also after the traverse, every node reachable from v in V_{i+1}, because of the flipping, can have different labels for the two adjacent edges. So the induced graph by these nodes reachable, becomes a permutation. Notice that for each next $v \in V_i$, we only do the traverse when it is not reachable from previous nodes in V_i. So each edge will be visited and may be flipped only once.

After we went through every $v \in V_i$, all nodes of V_{i+1} in S are reached. So we have a permutation. □

E Proof sketch for Lemma 4.6

Though not explicitly mentioned, the analysis of the INW generator in appendix B of [CHL+23] works for large alphabet ROBPs. Here we slightly modifies their proof to make it explicitly works.

We start with the definition of the INW generator. Some notions are defined in the preliminaries of Section 5

Definition E.1 (INW generator for large alphabets)**.** *Let n, c, d be powers of two. For each $t \in [\log n]$, let H_t be a c-regular bigraph $H_t = ([d \cdot c^{t-1}], [d \cdot c^t], E_{H_t})$, with a one-way label. Relative to the family $(H_t)_{t\in[\log n]}$, we recursively define the INW generator as follows:*

Define $\mathsf{INW}_0 : \{0,1\}^{\log d} \to \{0,1\}^{\log d}$ as the trivial PRG. For each $t \in [\log n]$, having defined $\mathsf{INW}_{t-1} : \{0,1\}^{d+(t-1)\log c} \to \{0,1\}^{\log d \cdot 2^{t-1}}$, we define $\mathsf{INW}_t : \{0,1\}^{d+t\log c} \to \{0,1\}^{\log d\cdot 2^t}$ as $\mathsf{INW}_t(x, y) = \mathsf{INW}_{t-1}(x), \mathsf{INW}_{t-1}(H_t[x, y])$. Here the comma denotes concatenation.

The INW generator relates to the derandomized product of permutation ROBPs, since they have consistent one-way labelings.

Definition E.2 (consistent consistent one-way labelings)**.** *Let $G = (U, V, E)$ be a d-regular bigraph. A consistent one-way labeling of G is a one-way labeling of G such that for every $v \in V$, the labels of the incoming edges of v are distinct, i.e., the $G[v, i] = G[u, i]$ implies $u = v$. In this case, we can extend the labeling to a two-way labelings:*

$$\mathsf{Rot}_G(u, i) = (G[u, i], i),$$

i.e., the incoming label and the outgoing label of an edge (u, v) are the same.

Lemma E.3 (Derandomized Product with consistent consistent one-way labelings,[RV05])**.** *Let $G_1 = (U, V, E_1)$ and $G_2 = (V, W, E_2)$ be two d-regular bigraphs with consistent one-way labelings. Let H be a x-regular bigraph with one-way labeling. Then $G_1 ⓟ_H G_2$ has a consistent one-way labeling.*

Let f be a permutation ROBP in $\mathcal{P}_{(n,s)}$. Then f is also a regular ROBP with a consistent labeling. It can be sv-approximated with the method of Lemma 5.10. Furthermore, the method corresponds to fooling f with the INW generator.

Lemma E.4 (equivariance between INW generator and Lemma 5.10)**.** *Let f be a permutation ROBP in $\mathsf{P}_{(n,s)}$. Let $t \in [\log n]$, $(j, j + 2^t) \in E_{SC_n}$ and $\widetilde{G}_{j,j+2^t}$ be the $(d \cdot c^t)$-regular bigraph with two-way labeling as defined in Lemma 5.10. Then for any $u \in V^j$ and $x \in \{0,1\}^{d+t\log c}$, we have*

$$\exists x', \widetilde{G}_{j,j+2^t}[u, x] = [v, x'] \Leftrightarrow \left[f^{[j,j+2^t]}(\mathsf{INW}(x))\right]_{u,x} = v,$$

Proof. The proof is by induction on t. For the base case $t = 0$, both sides mean u has an outgoing edge labeled x going to v. Assume the statement holds for $t - 1$. Recall that

$$\widetilde{G}_{j\to j+2^t} = \widetilde{G}_{j\to j+2^{t-1}} ⓟ_{H_t} \widetilde{G}_{j+2^{t-1}\to j+2^t}, \forall t \geq 1,$$

Then for t, the left side can be computed as follows (where $x = (z, y)$):

$$\begin{aligned}(w, z) &= \mathsf{Rot}_{\widetilde{G}_{j,j+2^{t-1}}}(u, z),\\ (z', y') &= \mathsf{Rot}_{H_t}(z, y),\\ (v, w) &= \mathsf{Rot}_{\widetilde{G}_{j+2^{t-1},j+2^t}}(z', y').\end{aligned}$$

Where the right side can be computed as follows:

$$\begin{aligned}
&f^{[j,j+2^{t-1}]}(\mathsf{INW}_{t-1}(z)) = w,\\
&H_t[z,y] = z',\\
&f^{[j+2^{t-1},j+2^t]}(\mathsf{INW}_{t-1}(z')) = v.
\end{aligned}$$

The induction hypothesis implies that the two sides are equivalent. □

Finally, we can prove the lemma.

Lemma E.5 (Lemma 4.6 restated)**.** *For all s, n, ε there exists a PRG (actually the INW generator) that fools $\mathcal{P}_{(n,s)}$ with ε-sv-error. The seed length is*

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

Proof. Let $\lambda = \min\{\varepsilon/(11\log n), 1/6(\log^2 n)\}$. Let $d = 2^s$ and for each $t \in [\log n]$, let H_t be a c-regular expanders with $\lambda(H_t) \le \lambda$ from Lemma 5.12. Let $\{\mathsf{INW}_t\}_{t\in[\log n]}$ be the INW generator from the definition. Let $\mathsf{INW} = \mathsf{INW}_{\log n}$. Then INW is a PRG with seed lengths

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

For any $f \in \mathcal{P}_{(n,s)}$, the derandomized product of f coincides with the the output of $f \circ INW$, i.e. $\widetilde{\mathbf{W}}_{n\leftarrow 0} = E_X f^{[0,n]}(\mathsf{INW}(X))$. Since $\widetilde{\mathbf{W}}_{n\leftarrow 0} \overset{sv}{\approx}_{11\lambda\log n} \widetilde{\mathbf{W}}_{n\leftarrow n-1}\cdots\widetilde{\mathbf{W}}_{1\leftarrow 0}$ by Lemma 5.10, INW fools f with ε-sv-error. □

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