Electronic Colloquium on Computational Complexity, Report No. 48 (2021)

Better Pseudodistributions and Derandomization for Space-Bounded Computation

William M. Hoza*
Department of Computer Science
University of Texas at Austin
whoza@utexas.edu

Abstract

Three decades ago, Nisan constructed an explicit pseudorandom generator (PRG) that fools width-n length-n read-once branching programs (ROBPs) with error ε and seed length $O(\log^2 n + \log n \cdot \log(1/\varepsilon))$ [Nis92]. Nisan's generator remains the best explicit PRG known for this important model of computation. However, a recent line of work starting with Braverman, Cohen, and Garg [BCG20; CL20; CDRSTS21; PV21] has shown how to construct *weighted* pseudorandom generators (WPRGs, aka pseudorandom pseudodistribution generators) with better seed lengths than Nisan's generator when the error parameter ε is small.

In this work, we present an explicit WPRG for width-n length-n ROBPs with seed length $O(\log^2 n + \log(1/\varepsilon))$. Our seed length eliminates $\log\log$ factors from prior constructions, and our generator completes this line of research in the sense that further improvements would require beating Nisan's generator in the standard constant-error regime. Our technique is a variation of a recently-discovered reduction that converts moderate-error PRGs into low-error WPRGs [CDRSTS21; PV21]. Our version of the reduction uses averaging samplers.

We also point out that as a consequence of the recent work on WPRGs, any randomized space-S decision algorithm can be simulated deterministically in space $O(S^{3/2}/\sqrt{\log S})$. This is a slight improvement over Saks and Zhou's celebrated $O(S^{3/2})$ bound [SZ99]. For this application, our improved WPRG is not necessary.

1 Introduction

1.1 Derandomization

Randomization is a versatile technique in algorithm design. However, random bits are not always available. Therefore, we would like to deterministically simulate randomized algorithms as efficiently as possible. In this paper, we focus on space efficiency. After fixing its input, the output of a small-space algorithm as a function of its random bits can be computed by a *read-once branching program* (ROBP).

Definition 1.1 (ROBP)**.** A width-w length-n *ROBP* is a directed graph consisting of $n+1$ layers of vertices $V_0, \ldots, V_n$ with w vertices in each layer. For each $i \in [n]$, each vertex in V_{i-1} has two outgoing edges labeled 0 and 1 leading to V_i. On input $x \in \{0,1\}^n$, the program starts at a designated start vertex $v_{\text{start}} \in V_0$, then reads the bits $x_1, \ldots, x_n$ in order and traverses the corresponding edges. The program accepts or rejects depending on whether the final vertex in

*Supported by the NSF GRFP under Grant DGE-1610403 and by a Harrington Fellowship from UT Austin.

ISSN 1433-8092

this path is a designated accept vertex $v_{\text{acc}} \in V_n$. In this way, the program computes a function $f\colon \{0,1\}^n \to \{0,1\}$.

Arguably, the most important case is $w = n$, which captures $(\log n)$-space randomized algorithms that always halt. To derandomize such an algorithm, we would like to estimate the expectation of the corresponding ROBP on a uniform random input.

1.2 Pseudorandom Generators

The traditional approach to derandomization is to design a *pseudorandom generator* (PRG).

Definition 1.2 (PRG)**.** Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$. An *$\varepsilon$-PRG* for $\mathcal{F}$ is a function $G\colon \{0,1\}^r \to \{0,1\}^n$ such that for every $f \in \mathcal{F}$,

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

Here r is the *seed length* of G.

By the probabilistic method, there exists a (nonexplicit) PRG for width-n length-n ROBPs with seed length $O(\log(n/\varepsilon))$. A corresponding explicit[1] construction would imply a complete derandomization of space-bounded computation ($\mathbf{L} = \mathbf{BPL}$), because we could deterministically estimate the expectation of a given ROBP f by computing $2^{-r} \cdot \sum_{x \in \{0,1\}^r} f(G(x))$. Babai, Nisan, and Szegedy designed the first explicit PRG for width-n length-n ROBPs [BNS92], with seed length

$$2^{O(\sqrt{\log n})} \cdot \log(1/\varepsilon).$$

In a subsequent breakthrough [Nis92], Nisan designed a PRG with a much better seed length of

$$O(\log^2 n + \log n \cdot \log(1/\varepsilon)).$$

1.3 Weighted PRGs

In the decades since Nisan's work [Nis92], despite intense effort, the problem of designing PRGs for width-n length-n ROBPs has stubbornly resisted further attacks. Nisan's PRG [Nis92] remains the best explicit PRG known for this model. However, PRGs are not the only possible approach to derandomization. Braverman, Cohen, and Garg recently introduced an intriguing generalization of PRGs called *weighted pseudorandom generators* (WPRGs), aka *pseudorandom pseudodistribution generators* [BCG20].

Definition 1.3 (WPRG)**.** Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$. An *$\varepsilon$-WPRG* for $\mathcal{F}$ is a pair of functions (G, ρ), where $G\colon \{0,1\}^r \to \{0,1\}^n$ and $\rho\colon \{0,1\}^r \to \mathbb{R}$, such that for every $f \in \mathcal{F}$,

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

Here r is the *seed length* of (G, ρ). If ρ maps $\{0,1\}^r \to [-K, K]$, we say the WPRG is *K-bounded*.

[1] We say that a function $G\colon \{0,1\}^r \to \{0,1\}^n$ is *explicit* if it can be computed in space $O(r)$. More precisely, we are considering a family of functions indexed by one or more parameters (e.g., n and ε). The algorithm for computing G is given both the parameters and the input to G.

A standard ("unweighted") PRG is the case $\rho(x) \equiv 1$. Just like an unweighted PRG, a WPRG for ROBPs can be used to estimate the expectation of a given ROBP f, because we can compute $2^{-r} \cdot \sum_{x \in \{0,1\}^r} \rho(x) \cdot f(G(x))$. As long as r is small and G and ρ are both efficiently computable, this is still an efficient derandomization. Thus, optimal WPRGs for ROBPs would immediately imply $\mathbf{L} = \mathbf{BPL}$. Furthermore, WPRGs imply *hitting set generators* (HSGs).

Definition 1.4 (HSG)**.** Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$. An *$\varepsilon$-HSG* for $\mathcal{F}$ is a function $G\colon \{0,1\}^r \to \{0,1\}^n$ such that for every $f \in \mathcal{F}$,

$$\mathop{\mathbb{E}}_{x \in \{0,1\}^n}[f(x)] \geq \varepsilon \implies \exists x \in \{0,1\}^r,\ f(G(x)) = 1.$$

If (G, ρ) is an ε-WPRG for $\mathcal{F}$, then G is an ε'-HSG for $\mathcal{F}$ for any $\varepsilon' > \varepsilon$ [BCG20]. HSGs have been studied since the 80s [AKS87], but prior to Braverman, Cohen, and Garg's work [BCG20], no explicit HSG for width-n length-n ROBPs was known that was any better than Nisan's PRG (except when ε is *extremely* small; see Fig. 1). For these reasons, it was exciting when Braverman, Cohen, and Garg presented an explicit WPRG for width-n length-n ROBPs [BCG20] with seed length

$$\widetilde{O}(\log^2 n + \log(1/\varepsilon)),$$

which is better than Nisan's PRG's seed length when $\varepsilon \ll 1/\operatorname{poly}(n)$.

Admittedly, Braverman, Cohen, and Garg's result [BCG20] did not yet imply an improved derandomization of space-bounded computation, but still, their innovative and complex work provides valuable insights. The additional flexibility in the definition of a WPRG means that WPRGs can be easier to construct compared to unweighted PRGs. In fact, in one setting (unbounded-width permutation ROBPs with a single accept vertex), Pyne and Vadhan recently showed that there is an explicit WPRG [PV21] with a seed length that is *provably impossible* to attain by unweighted PRGs [HPV21], a testament to the power of the WPRG approach to derandomization.

Subsequent to Braverman, Cohen, and Garg's work [BCG20], Chattopadhyay and Liao gave a simpler WPRG construction [CL20] for width-n length-n ROBPs with the improved seed length

$$\widetilde{O}(\log^2 n) + O(\log(1/\varepsilon)). \tag{1}$$

Very recently, Cohen, Doron, Renard, Sberlo, and Ta-Shma [CDRSTS21] and Pyne and Vadhan [PV21] independently obtained an even simpler WPRG for width-n length-n ROBPs with seed length

$$O(\log^2 n) + \widetilde{O}(\log(1/\varepsilon)). \tag{2}$$

(These last two constructions and analyses are nearly identical [CDRSTS21; PV21].)

1.4 Main Result: An Improved WPRG

In this work, we present another WPRG for ROBPs with a better seed length.

Theorem 1.5. *For any $w, n \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit ε-WPRG for width-w length-n ROBPs with seed length $O(\log(wn)\log n + \log(1/\varepsilon))$. Furthermore, the WPRG is $\operatorname{poly}(1/\varepsilon)$-bounded.*

When $w = n$, our WPRG has seed length $O(\log^2 n + \log(1/\varepsilon))$, giving the "best of both worlds" compared to Eqs. (1) and (2). Our WPRG is the first to achieve seed length $O(\log^2 n)$ with error $n^{-\log n}$. Furthermore, our WPRG represents the completion of the research project of designing WPRGs for width-n length-n ROBPs while focusing on the seed length's dependence on ε [BCG20; CL20; CDRSTS21; PV21]. After all, even an HSG must have seed length at least $\Omega(\log(1/\varepsilon))$, so

Seed length	Type of generator	Reference
$\widetilde{O}(\sqrt{n}) + O(\log(1/\varepsilon))$	HSG	[AKS87][2]
$2^{O(\sqrt{\log n})} \cdot \log(1/\varepsilon)$	PRG	[BNS92]
$O(\log^2 n + \log(1/\varepsilon) \cdot \log n)$	PRG	[Nis92]
$\widetilde{O}(\log^2 n + \log(1/\varepsilon))$	WPRG	[BCG20]
$O(\log^2 n + \log(1/\varepsilon))$	HSG	[HZ20]
$\widetilde{O}(\log^2 n) + O(\log(1/\varepsilon))$	WPRG	[CL20]
$O(\log^2 n) + \widetilde{O}(\log(1/\varepsilon))$	WPRG	[CDRSTS21; PV21]
$O(\log^2 n + \log(1/\varepsilon))$	WPRG	This work
$O(\log n + \log(1/\varepsilon))$	PRG	Optimal (non-explicit)

Figure 1: Known PRGs, WPRGs, and HSGs for width-n length-n ROBPs. As a reminder, PRG $\implies$ WPRG $\implies$ HSG.

obtaining a better WPRG for width-n length-n ROBPs requires beating Nisan's generator in the traditional, challenging constant-error regime. (That being said, see Section 6.)

Our WPRG generalizes some other recent work on the small-ε regime. Hoza and Zuckerman constructed an explicit ε-HSG for width-n length-n ROBPs with seed length $O(\log^2 n + \log(1/\varepsilon))$ [HZ20], which follows also from our WPRG. Meanwhile, Cheng and Hoza gave a deterministic algorithm for estimating $\mathbb{E}[f] \pm \varepsilon$ in space $O(\log^2 n + \log(1/\varepsilon))$ given query access to a constant-width ROBP f [CH20]; Theorem 1.5 immediately implies such an algorithm for the more general case of polynomial-width ROBPs.

1.5 Derandomization that Beats the Saks-Zhou Bound

Next we turn to the general problem of derandomizing space-S decision algorithms, whether by PRGs, WPRGs, HSGs, or any other method. Early work [Sav70; Jun81; BCP83] implies that such algorithms can be simulated deterministically in space $O(S^2)$. Saks and Zhou gave an improved algorithm that runs in space $O(S^{3/2})$ [SZ99], which has remained unbeaten for decades. We point out that as a consequence of the recent progress on WPRGs, it is now possible to slightly improve the bound.

Theorem 1.6. *For any space-constructible function $S(N) \geq \log N$, we have*

$$\mathbf{BPSPACE}(S) \subseteq \mathbf{DSPACE}\left(\frac{S^{3/2}}{\sqrt{\log S}}\right).$$

(We use N to denote the input length, reserving n to denote the length of an ROBP. Recall that $\mathbf{BPSPACE}(S)$ is the class of languages that can be decided by randomized algorithms that run

[2]For any $w \in \mathbb{N}$, Ajtai, Komlos, and Szemeredi designed an explicit $(1/w)$-HSG for width-w length-n ROBPs where $n = O(\log^2 w / \log\log w)$ with optimal seed length $O(\log w)$ [AKS87]. Turning things around, for any $n \in \mathbb{N}$ and $\varepsilon > 0$, we can let $w = 2^{\sqrt{n \log n}}/\varepsilon$ and get an explicit ε-HSG for width-w length-n ROBPs (hence also for width-n length-n ROBPs) with seed length $O(\sqrt{n \log n} + \log(1/\varepsilon))$.

in space $O(S)$ and always halt.) Admittedly, $O(S^{3/2}/\sqrt{\log S})$ is barely any better than Saks and Zhou's $O(S^{3/2})$ bound [SZ99]. However, we hope that Theorem 1.6 might break a "psychological barrier" by demonstrating that the Saks-Zhou algorithm [SZ99] has room for improvement.

Our improved WPRG is not necessary for proving Theorem 1.6. Instead, Theorem 1.6 follows by combining several prior works [SZ99; Arm98; KNW08; CL20; CDRSTS21; PV21].

1.6 Overview of Proofs

1.6.1 Overview of our Improved WPRG

The proof of Theorem 1.5 is similar to the recent WPRG constructions by Cohen et al. and Pyne and Vadhan [CDRSTS21; PV21]. Say we would like to fool some width-n length-n ROBP f with low error $\varepsilon \ll 1/\operatorname{poly}(n)$. The starting point is a PRG G that fools ROBPs with moderate error $1/\operatorname{poly}(n)$. Building on work by Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan [AKMPSV20], Cohen et al. and Pyne and Vadhan [CDRSTS21; PV21] showed a bound of the form

$$\left|\mathbb{E}[f] - \sum_{i=1}^{K} \sigma_i \cdot \mathbb{E}[f(A_i)]\right| \leq \varepsilon, \tag{3}$$

where $K = \operatorname{poly}(1/\varepsilon)$, each $\sigma_i = \pm 1$, and each random variable A_i is a concatenation of $O\left(\frac{\log(1/\varepsilon)}{\log n}\right)$ truncations of independent samples from G. From here, we could immediately obtain an ε-WPRG by taking G to be Nisan's generator [Nis92], but such a WPRG would have seed length $\Omega(\log(1/\varepsilon) \cdot \log n)$ due to the cost of sampling A_i via independent seeds to Nisan's generator. To get a better seed length, we would like to use *correlated* seeds to Nisan's generator.

The approach of Cohen et al. and Pyne and Vadhan [CDRSTS21; PV21] is to use the Impagliazzo-Nisan-Wigderson (INW) PRG [INW94] to generate a pseudorandom sequence of seeds to Nisan's generator. Because the INW generator is non-optimal, this approach leads to the seed length $O(\log^2 n + \log(1/\varepsilon) \log \log_n(1/\varepsilon))$.

Our approach is based on a simple observation. The proof of Eq. (3) does not actually require that G fool *all* width-n length-n ROBPs. Indeed, Eq. (3) holds under the weaker assumption that G fools all subprograms of *the specific ROBP* f that we are analyzing.

To exploit this observation, we apply a trick that uses an "averaging sampler" Samp. We start with a PRG G_0 for width-n length-n ROBPs with moderate error $1/\operatorname{poly}(n)$ and seed length $O(\log^2 n)$, such as Nisan's generator [Nis92]. Our WPRG selects a string x of length $O(\log^2 n + \log(1/\varepsilon))$ uniformly at random. The sampler condition implies that for any ROBP f, with high probability over x, the PRG $G(y) \stackrel{\text{def}}{=} G_0(\mathsf{Samp}(x, y))$ fools all subprograms of f with error $1/\operatorname{poly}(n)$ and optimal seed length $O(\log n)$. Our WPRG now applies Eq. (3) to G rather than G_0. Because G has such a short seed length, sampling A_i only costs us $O(\log(1/\varepsilon))$ truly random bits now, which we can afford. (Similar tricks have been used previously in space-bounded derandomization [Nis94; Arm98; HZ20].)

In general, our reduction converts any PRG for width-w length-n ROBPs with error $1/\operatorname{poly}(wn)$ and seed length r into a WPRG for width-w length-n ROBPs with any desired error ε and seed length $O(r + \log(wn/\varepsilon))$.

We also take this opportunity to give a slightly different perspective on the proof of Eq. (3). The original proof of Eq. (3) is based on "preconditioned Richardson iteration," a method for improving the accuracy of an approximate matrix inverse [AKMPSV20; CDRSTS21; PV21]. Cohen et al. pointed out that the proof has a resemblance to the notion of *local consistency errors* introduced by Cheng and Hoza [CH20], and indeed, we show that Eq. (3) can be understood entirely in terms

of local consistency. As we explain in Appendix A, this is not a substantially different proof, but rather a different way of thinking about the same proof. We hope that this alternative perspective might yield new insights in the future.

1.6.2 Overview of our Improved Derandomization

The proof of Theorem 1.6 (simulating randomized space S in deterministic space $o(S^{3/2})$) builds on Saks and Zhou's algorithm [SZ99]. To derandomize space-$(\log w)$ algorithms, Saks and Zhou rely heavily on Nisan's PRG for width-w length-n ROBPs. Crucially, Saks and Zhou set n to be *much smaller* than w. To fool such programs with error ε, Nisan's PRG has seed length

$$O(\log(wn/\varepsilon)\log n),$$

so by choosing $n = 2^{O(\sqrt{\log w})}$ and $\varepsilon = 1/\operatorname{poly}(w)$, the seed length of Nisan's PRG is only $O(\log^{3/2} w)$. The crux of Saks and Zhou's work [SZ99] is a clever method of reusing a seed of this PRG many times to derandomize a $(\log w)$-space algorithm even though it might use up to w random bits.

Saks and Zhou's work therefore provides extra motivation for studying width-w length-n ROBPs when $n \ll w$. These programs correspond to algorithms that only use a small amount of randomness. In this "low-randomness" regime, PRGs have long been known that are slightly better than Nisan's PRG. Most famously, Nisan and Zuckerman designed a PRG for the case $n = \operatorname{polylog} w$ with error $2^{-\log^{0.99} w}$ and optimal seed length $O(\log w)$ [NZ96]. Later, Armoni designed a PRG that interpolates between Nisan's PRG [Nis92] and the Nisan-Zuckerman PRG [NZ96], suitable for the regime $\operatorname{polylog} w \ll n \ll w$ [Arm98]. Using extractors that were not available to Armoni at the time of his work [Arm98], Armoni's PRG can be implemented [KNW08] to have seed length

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

For $n \ll w$ and $\varepsilon = 1/\operatorname{poly}(n)$, this is better than Nisan's PRG by a factor of $\Theta(\log\log w)$.

Furthermore, although Saks and Zhou [SZ99] relied on the specific structure of Nisan's PRG [Nis92], Armoni showed how to modify the Saks-Zhou algorithm to use any generic PRG for ROBPs [Arm98]. It is therefore natural to try to improve the Saks-Zhou theorem by replacing Nisan's PRG with Armoni's, and indeed, it has been suggested that Theorem 1.6 follows already from Armoni's work.[3]

However, it seems that Theorem 1.6 does *not* follow directly from Armoni's work. The trouble is the error parameter. For the Saks-Zhou method to work, it seems to be necessary that the PRG has error $1/\operatorname{poly}(w)$ rather than $1/\operatorname{poly}(n)$. When $\varepsilon = 1/\operatorname{poly}(w)$, Armoni's PRG is no better than Nisan's PRG, so we get no improvement. Armoni himself understood this issue and did not claim to beat the Saks-Zhou bound in the general case. Instead, he showed how to use his PRG to get an improved derandomization of *low-randomness* algorithms [Arm98].

Today, however, we have new tools for fooling ROBPs with low error. In particular, we can use the recent error reduction procedure due to Cohen et al. and Pyne and Vadhan [CDRSTS21; PV21]. Cohen et al. show how to convert a PRG for width-w length-n ROBPs with error $1/\operatorname{poly}(n)$[4] and seed length r into a WPRG for width-w length-n ROBPs with any desired error ε and seed length

[3] I have heard a speaker make this claim during an oral presentation, but the speaker clarified that they were not familiar with a careful proof and were merely communicating what someone else had said. I am also aware of an instance in which this claim was made in typeset lecture notes, but the claim was removed after a revision.

[4] Pyne and Vadhan's analysis [PV21] is slightly looser; they assume that the initial PRG has error $1/\operatorname{poly}(wn)$ (just like we do in our WPRG construction), which is too small for proving Theorem 1.6.

$r + \widetilde{O}(\log(w/\varepsilon))$ [CDRSTS21]. Applying this reduction to Armoni's PRG with $n = 2^{\sqrt{\log w \cdot \log\log w}}$ (slightly larger than the choice in Saks and Zhou's original work [SZ99]), we obtain a WPRG for width-w length-n ROBPs with error $1/\operatorname{poly}(w)$ and seed length

$$O\left(\frac{\log^{3/2} w}{\sqrt{\log\log w}}\right) + \widetilde{O}(\log w) = O\left(\frac{\log^{3/2} w}{\sqrt{\log\log w}}\right).$$

Meanwhile, Chattopadhyay and Liao showed [CL20] that WPRGs can be used in place of PRGs in Saks and Zhou's algorithm, provided the WPRG is $\operatorname{poly}(w)$-bounded. The WPRG from Cohen et al.'s reduction [CDRSTS21] is indeed $\operatorname{poly}(1/\varepsilon)$-bounded, completing the proof of Theorem 1.6.

1.7 WPRGs vs. HSGs

We remark that the proof of Theorem 1.6 sheds light on the relative strengths of HSGs and WPRGs. Cheng and Hoza recently showed that optimal HSGs would imply $\mathbf{L} = \mathbf{BPL}$ [CH20], which might cause one to question whether WPRGs have value above and beyond the value of HSGs. Chattopadhyay and Liao addressed this concern by showing that WPRGs could hypothetically be used in the Saks-Zhou algorithm to prove a new and improved derandomization of space-bounded computation [CL20], whereas it is still not known how to use HSGs in the Saks-Zhou algorithm. Theorem 1.6 makes the hypothetical possibility envisioned by Chattopadhyay and Liao a reality[5] and thereby demonstrates the strength of the WPRG approach to derandomization.

2 Preliminaries

2.1 Pseudodistributions

For most of our analysis, we will work with *pseudodistributions* rather than the WPRG formalism. For our puposes, a pseudodistribution is a generalization of a probability distribution in which probabilities are replaced with "pseudoprobabilities," which are arbitrary real numbers that do not necessarily sum to one.

Definition 2.1 (Pseudodistribution)**.** A *pseudodistribution* over $\{0,1\}^n$ is a formal real linear combination of n-bit strings,[6] i.e., a sum of the form

$$A = \sum_{i=1}^{R} a_i \cdot x^{(i)},$$

where $a_i \in \mathbb{R}$ and $x^{(i)} \in \{0,1\}^n$. A probability distribution over $\{0,1\}^n$ is the special case that $a_i \geq 0$ and $\sum_{i=1}^{R} a_i = 1$. We define U_n to be the uniform distribution over $\{0,1\}^n$, i.e.,

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

[5] To be clear, we only achieve derandomization in space $O(S^{3/2}/\sqrt{\log S})$, whereas Chattopadhyay and Liao proposed a route toward the much better bound $O(S^{4/3})$, developing an earlier proposal by Braverman, Cohen, and Garg [BCG20].

[6] Equivalently, A is a vector in the n-fold tensor product space $\mathbb{R}^2 \otimes \cdots \otimes \mathbb{R}^2 \cong \mathbb{R}^{2^n}$. The reader might find it helpful to make an analogy with quantum computing; recall that a pure state of an n-qubit system is a vector in the n-fold tensor product space $\mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2 \cong \mathbb{C}^{2^n}$. We could even have used ket notation for pseudodistributions: $A = \sum_{i=1}^{R} a_i \cdot |x^{(i)}\rangle$.

We often identify a function f on $\{0,1\}^n$ with the induced probability distribution $f(U_n)$. We define the *pseudoexpectation* of a function $f\colon \{0,1\}^n \to \mathbb{R}$ under the pseudodistribution A by

$$\widetilde{\mathbb{E}}[f(A)] = \sum_{i=1}^{R} a_i \cdot f(x^{(i)}).$$

We say that A *fools* f with error ε if

$$\left|\mathbb{E}[f] - \widetilde{\mathbb{E}}[f(A)]\right| \leq \varepsilon.$$

Definition 2.2 (Operations on Pseudodistributions)**.** *Linear combinations* of pseudodistributions over $\{0,1\}^n$ are defined in the natural way. The *tensor product* of two pseudodistributions is given by

$$\left(\sum_{i=1}^{R} a_i \cdot x^{(i)}\right) \otimes \left(\sum_{j=1}^{R'} b_j \cdot y^{(j)}\right) = \sum_{i=1}^{R}\sum_{j=1}^{R'} a_i b_j \cdot (x^{(i)} \circ y^{(j)}),$$

where $\circ$ denotes concatenation. Thus if A is a pseudodistribution over $\{0,1\}^n$ and B is a pseudodistribution over $\{0,1\}^{n'}$, then $A \otimes B$ is a pseudodistribution over $\{0,1\}^{n+n'}$.

2.2 Weighted PRGs

As discussed in Section 1, a WPRG is a pair (G,ρ), where $G\colon \{0,1\}^r \to \{0,1\}^n$ and $\rho\colon \{0,1\}^r \to \mathbb{R}$. Each WPRG has a corresponding pseudodistribution, just as a PRG has a corresponding distribution.

Definition 2.3 (Pseudodistribution Sampled by a WPRG)**.** If (G,ρ) is a WPRG with seed length r, the pseudodistribution *sampled by* (G,ρ) is

$$A = \sum_{x \in \{0,1\}^r} 2^{-r} \cdot \rho(x) \cdot G(x).$$

Note that (G,ρ) is an ε-WPRG for f if and only if A fools f with error ε.

WPRGs can be combined in the same ways as pseudodistributions. Consideration of these operations will help us verify the seed length, boundedness, and efficiency of our WPRG.

Definition 2.4 (Operations on WPRGs)**.** Suppose we have two WPRGs (G_0,ρ_0) and (G_1,ρ_1), where $G_b\colon \{0,1\}^{r_b} \to \{0,1\}^{n_b}$ and $\rho_b\colon \{0,1\}^{r_b} \to \mathbb{R}$. We define the tensor product $(G_0,\rho_0) \otimes (G_1,\rho_1)$ to be a WPRG (G,ρ) with seed length $r_0 + r_1$ given by

$$\begin{aligned} G(x,y) &= G_0(x) \circ G_1(y) \\ \rho(x,y) &= \rho_0(x) \cdot \rho_1(y). \end{aligned}$$

If $n_0 = n_1$, we define the sum $(G_0,\rho_0) + (G_1,\rho_1)$ to be a WPRG (G,ρ) with seed length $r+1$ given by

$$\begin{aligned} G(x,b) &= G_b(x) \\ \rho(x,b) &= 2 \cdot \rho_b(x). \end{aligned}$$

(There is a factor of 2 because in the definition of WPRGs, we look at an expectation over seeds rather than a sum.) For a WPRG (G,ρ) and a real number c, we define $c \cdot (G,\rho) = (G,\rho')$, where

$$\rho'(x) = c \cdot \rho(x).$$

Under these definitions, if (G_b, ρ_b) samples from the pseudodistribution A_b over $\{0,1\}^{n_b}$, then $(G_0, \rho_0) \otimes (G_1, \rho_1)$ samples from $A_0 \otimes A_1$, and if $n_0 = n_1$, then $(G_0, \rho_0) + c \cdot (G_1, \rho_1)$ samples from $A_0 + cA_1$. Furthermore, if (G_b, ρ_b) is K_b-bounded, then $(G_0, \rho_0) \otimes (G_1, \rho_1)$ is (K_0K_1)-bounded; if (G_0, ρ_0) and (G_1, ρ_1) are both K-bounded, then $(G_0, \rho_0) + (G_1, \rho_1)$ is $(2K)$-bounded; if (G, ρ) is K-bounded, then $c \cdot (G, \rho)$ is (cK)-bounded.

2.3 Applying Pseudodistributions to ROBPs

Let f be an ROBP with layers $V_0, \dots, V_n$. Let $u \in V_i$ and $v \in V_j$. When $j \geq i$, we define the *subprogram* $f_{u \to v} \colon \{0,1\}^{j-i} \to \{0,1\}$ to be the length-$(j-i)$ ROBP obtained from f by setting u to be the start vertex and v to be the accept vertex. For convenience, we extend $f_{u \to v}$ to a function $f_{u \to v} \colon \{0,1\}^{\geq j-i}$ that ignores all but the first $j - i$ bits of its input.

If A is a pseudodistribution over $\{0,1\}^d$ with $i + d \geq j$, we define $A[u \to v]$ to be the pseudoprobability of reaching v from u using A, i.e.,

$$A[u \to v] = \widetilde{\mathbb{E}}[f_{u \to v}(A)].$$

We extend the definition by defining $A[u \to v] = 0$ when $i > j$.

2.4 Local Consistency

As mentioned in Section 1.6.1, we will present a WPRG analysis based on the notion of *local consistency* introduced by Cheng and Hoza [CH20]. The idea behind local consistency is that we measure the quality of a pseudodistribution by using it to estimate $\mathbb{E}[f_{u \to v}]$ in two different ways and comparing the results.

Definition 2.5 (Local Consistency Error)**.** Let f be an ROBP with layers $V_0, \dots, V_n$. Let $u \in V_i$ and $v \in V_j$ with $i < j$, and let A be a pseudodistribution over $\{0,1\}^d$ with $i + d \geq j$. The *local consistency error* $\mathrm{LCErr}_{u \to v}(A)$ is defined by

$$\mathrm{LCErr}_{u \to v}(A) = \left(\sum_{t \in V_{j-1}} A[u \to t] \cdot U_1[t \to v] \right) - A[u \to v].$$

We extend the definition by setting $\mathrm{LCErr}_{u \to v}(A) = 0$ when $j \leq i$. We say that A is *α-locally consistent* with respect to f if for every u, v we have $|\mathrm{LCErr}_{u \to v}(A)| \leq \alpha$.

Note that U_n is 0-locally consistent. As we explain in Appendix A, local consistency is closely connected to approximating the inverse of the random walk Laplacian matrix of f. Cheng and Hoza's work [CH20] shows that local consistency and fooling are equivalent, up to some loss in the error parameter [CH20]. We repeat the argument here for clarity.

Lemma 2.6. *Let A be a pseudodistribution over $\{0,1\}^n$ and let f be a width-w length-n ROBP.*

1. *If A fools every subprogram $f_{u \to v}$ of f with error α, then A is $(2w\alpha)$-locally consistent with respect to f.*
2. *If A is ε-locally consistent with respect to f, then A fools every subprogram $f_{u \to v}$ of f with error $wn\varepsilon$.*

Proof. First, suppose A fools every subprogram $f_{u\to v}$ with error α. Then if $u \in V_i$ and $v \in V_j$ with $i < j$, we have

$$\begin{aligned}
|\operatorname{LCErr}_{u\to v}(A)| &= \left| A[u \to v] - \sum_{t \in V_{j-1}} A[u \to t] \cdot U_1[t \to v] \right| \\
&\le |A[u \to v] - U_n[u \to v]| + \left| U_n[u \to v] - \sum_{t \in V_{j-1}} A[u \to t] \cdot U_1[t \to v] \right| \\
&\le \alpha + \sum_{t \in V_{j-1}} |U_n[u \to t] - A[u \to t]| \cdot U_1[t \to v] \\
&\le (w+1)\alpha \le 2w\alpha.
\end{aligned}$$

Conversely, suppose A is ε-locally consistent with respect to f. Then for any $u \in V_i$ and any $j > i$,

$$\begin{aligned}
\sum_{v \in V_j} |A[u \to v] - U_n[u \to v]| &\le \sum_{v \in V_j} \left(\left| \sum_{t \in V_{j-1}} A[u \to t] U_1[t \to v] - U_n[u \to v] \right| + \varepsilon \right) \\
&\le w\varepsilon + \sum_{v \in V_j} \sum_{t \in V_{j-1}} |A[u \to t] - U_n[u \to t]| \cdot U_1[t \to v] \\
&= w\varepsilon + \sum_{t \in V_{j-1}} |A[u \to t] - U_n[u \to t]| \cdot \sum_{v \in V_j} U_1[t \to v] \\
&= w\varepsilon + \sum_{t \in V_{j-1}} |A[u \to t] - U_n[u \to t]|.
\end{aligned}$$

By induction on $j - i$, it follows that

$$\sum_{v \in V_j} |A[u \to v] - U_n[u \to v]| \le wn\varepsilon,$$

and hence A fools every subprogram with error $wn\varepsilon$. □

We remark that there is a version of Lemma 2.6 that eliminates both factors of w. To obtain such bounds, one can consider the *sum* over all $v \in V_j$ of each type of error $u \to v$. We have no need of this more refined analysis, so we omit the details.

3 Amplifying Local Consistency

Let G be a given pseudodistribution over $\{0,1\}^n$. (Ultimately we will take G to be a probability distribution, but this stage of the construction makes sense in the more general setting of pseudodistributions.) We will show how to combine multiple samples from G to *improve its local consistency*. Throughout this section, fix a length-n ROBP f with layers $V = V_0 \cup \cdots \cup V_n$.

3.1 Construction

For each $d \le n$, define G_d to be the pseudodistribution obtained by drawing a sample from G and truncating to the first d bits. That is, if $G = \sum_{i=1}^{R} a_i \cdot x^{(i)}$, then

$$G_d = \sum_{i=1}^{R} a_i \cdot x^{(i)}_{1\ldots d}. \tag{4}$$

Define a pseudodistribution Δ_d over $\{0,1\}^d$ by letting $\Delta_0 = 0$ and

$$\Delta_{d+1} = G_d \otimes U_1 - G_{d+1}.$$

The definition of Δ_d should remind the reader of local consistency errors. (See Lemma 3.2.) Now we define a "multi-hop" generalization of Δ_d. For $m \geq 0$, define a pseudodistribution $\Delta_d^{(m)}$ over $\{0,1\}^d$ by

$$\Delta_d^{(m)} = \sum_{d_1+\cdots+d_m=d} \Delta_{d_1} \otimes \cdots \otimes \Delta_{d_m},$$

where the sum is over all tuples of nonnegative integers $(d_1, \ldots, d_m)$ that sum to d. Note that $\Delta_0^{(0)}$ is the trivial probability distribution that assigns probability 1 to the empty string. Next, for each m, define a pseudodistribution $T^{(m)}$ over $\{0,1\}^n$ by

$$T^{(m)} = \sum_{d=0}^{n} \Delta_d^{(m)} \otimes G_{n-d},$$

and finally define a pseudodistribution $G^{(m)}$ over $\{0,1\}^n$ by

$$G^{(m)} = \sum_{i=0}^{m} T^{(i)} = \sum_{i=0}^{m} \sum_{d=0}^{n} \Delta_d^{(i)} \otimes G_{n-d}.$$

Note that $G^{(0)} = T^{(0)} = G$. We will show that as m gets large, $G^{(m)}$ becomes increasingly locally consistent.

3.2 Analysis

Define a "multi-hop" generalization of local consistency errors by

$$\mathrm{LCErr}^{(m)}_{u \to v}(G) = \sum_{u=u_0,u_1,\ldots,u_m=v} \prod_{j=1}^{m} \mathrm{LCErr}_{u_{i-1} \to u_i}(G),$$

where the sum is over all sequences of $m+1$ vertices starting with u and ending with v. Note the edge case $\mathrm{LCErr}^{(0)}_{u \to u} = 1$. Our goal in this section is to prove the following *exact formula* for the local consistency errors of $G^{(m)}$ in terms of the local consistency errors of G.

Lemma 3.1. *For any two vertices u, v and any $m \geq 0$, we have* $\mathrm{LCErr}_{u \to v}(G^{(m)}) = \mathrm{LCErr}^{(m+1)}_{u \to v}(G)$.

Let us briefly pause to marvel at this phenomenon. In most settings, when several imperfect ingredients are combined, we expect that the errors will compound on each other, so the combination has more error than any individual ingredient. We typically consider ourselves lucky if we can prove that the errors compund mildly. The remarkable feature of Lemma 3.1 is that it involves *products* of errors, i.e., the local consistency errors of G are actually combining *in our favor*!

Toward proving Lemma 3.1, we begin by analyzing Δ_d. It is immediate from the definitions that if $u \in V_i$ and $v \in V_{i+d}$, then $\Delta_d[u \to v] = \mathrm{LCErr}_{u \to v}(G)$. More generally, we have the following formula.

Lemma 3.2. *Let $i, j, d \leq n$. Let $u \in V_i$ and $v \in V_j$ and let A be a pseudodistribution over $\{0,1\}^{n-d}$. Then*

$$(\Delta_d \otimes A)[u \to v] = \sum_{t \in V_{i+d}} \mathrm{LCErr}_{u \to t}(G) \cdot A[t \to v]. \tag{5}$$

Proof. For the first case, suppose $i + d \leq j$. Then for any $x \in \{0,1\}^d$ and any $y \in \{0,1\}^{n-d}$, we have $f_{u\to v}(x,y) = \sum_{t \in V_{i+d}} f_{u\to t}(x) \cdot f_{t\to v}(y)$. Therefore, for any pseudodistribution B over $\{0,1\}^d$ whatsoever, we have

$$(B \otimes A)[u \to v] = \sum_{t \in V_{i+d}} B[u \to t] \cdot A[t \to v].$$

Since $\Delta_d[u \to t] = \mathrm{LCErr}_{u\to t}(G)$, we are done in this case.

For the second case, suppose $i+d > j$. Then either $i > j$, or else $d > 0$ and the pseudodistributions $G_{d-1} \otimes U_1 \otimes A$ and $G_d \otimes A$ agree on their first $j - i$ bits.[7] Either way, $(\Delta_d \otimes A)[u \to v] = 0$. Meanwhile, for each $t \in V_{i+d}$, trivially $A[t \to v] = 0$, so both sides of Eq. (5) are zero in this case. □

More generally, we have the following relationship between $\Delta_d^{(m)}$ and $\mathrm{LCErr}^{(m)}$.

Lemma 3.3. *Let $i, j \leq n$ and $m \geq 0$. Let $u \in V_i$ and $v \in V_j$ and let A be a pseudodistribution over $\{0,1\}^{n-d}$. Then for any $d \leq n$,*

$$(\Delta_d^{(m)} \otimes A)[u \to v] = \sum_{t \in V_{i+d}} \mathrm{LCErr}_{u\to t}^{(m)}(G) \cdot A[t \to v].$$

Proof. The base case $m = 0$ is the trivial statement $A[u \to v] = A[u \to v]$. For the inductive step, note that

$$\Delta_d^{(m+1)} = \sum_{k=0}^{d} \Delta_{d-k}^{(m)} \otimes \Delta_k,$$

so

$$\begin{aligned}
(\Delta_d^{(m+1)} \otimes A)[u \to v] &= \sum_{k=0}^{d} (\Delta_{d-k}^{(m)} \otimes \Delta_k \otimes A)[u \to v] && \text{(Linearity)} \\
&= \sum_{k=0}^{d} \sum_{s \in V_{i+d-k}} \mathrm{LCErr}_{u\to s}^{(m)}(G) \cdot (\Delta_k \otimes A)[u \to v] && \text{(Induction)} \\
&= \sum_{k=0}^{d} \sum_{s \in V_{i+d-k}} \sum_{t \in V_{i+d}} \mathrm{LCErr}_{u\to s}^{(m)}(G) \cdot \mathrm{LCErr}_{s\to t}(G) \cdot A[t \to v] && \text{(Lemma 3.2)} \\
&= \sum_{t \in V_{i+d}} \mathrm{LCErr}_{u\to t}^{(m+1)}(G) \cdot A[t \to v]. &&
\end{aligned}$$

□

Proof of Lemma 3.1. For any $u \in V_j$ and $v \in V_k$, by Lemma 3.3,

$$\begin{aligned}
T^{(m)}[u \to v] &= \sum_{d=0}^{n} \sum_{t \in V_{j+d}} \mathrm{LCErr}_{u\to t}^{(m)}(G) \cdot G[t \to v] \\
&= \sum_{t \in V} \mathrm{LCErr}_{u\to t}^{(m)}(G) \cdot G[t \to v].
\end{aligned}$$

[7] I.e., the induced pseudodistributions on the first $j - i$ bits (see Eq. (4)) are identical.

Therefore, if $u \in V_j$ and $v \in V_k$ with $j < k$, then

$$\begin{aligned}
\mathrm{LCErr}_{u\to v}(T^{(m)}) &= \left(\sum_{s\in V_{k-1}} T^{(m)}[u \to t] \cdot U_1[t \to v]\right) - T^{(m)}[u \to v] \\
&= \left(\sum_{s\in V_{k-1}} \sum_{t\in V} \mathrm{LCErr}^{(m)}_{u\to t}(G) \cdot G[t \to s] \cdot U_1[s \to v]\right) - \sum_{t\in V} \mathrm{LCErr}^{(m)}_{u\to t}(G) \cdot G[t \to v] \\
&= \sum_{t\in V} \mathrm{LCErr}^{(m)}_{u\to t}(G) \cdot \underbrace{\left(\left(\sum_{s\in V_{k-1}} G[t \to s] \cdot U_1[s \to v]\right) - G[t \to v]\right)}_{(*)}.
\end{aligned}$$

Quantity $(*)$ is exactly the local consistency error $\mathrm{LCErr}_{t\to v}(G)$, except in one edge case: when $t = v$, $\mathrm{LCErr}_{t\to t}(G) = 0$, whereas $(*) = -1$. Therefore,

$$\begin{aligned}
\mathrm{LCErr}_{u\to v}(T^{(m)}) &= \left(\sum_{t\in V} \mathrm{LCErr}^{(m)}_{u\to t}(G) \cdot \mathrm{LCErr}_{t\to v}(G)\right) - \mathrm{LCErr}^{(m)}_{u\to v}(G) \\
&= \mathrm{LCErr}^{(m+1)}_{u\to v}(G) - \mathrm{LCErr}^{(m)}_{u\to v}(G).
\end{aligned}$$

Thus, we get a telescoping sum:

$$\begin{aligned}
\mathrm{LCErr}_{u\to v}(G^{(m)}) &= \sum_{i=0}^{m} \mathrm{LCErr}_{u\to v}(T^{(i)}) \\
&= \sum_{i=0}^{m} \mathrm{LCErr}^{(i+1)}_{u\to v}(G) - \mathrm{LCErr}^{(i)}_{u\to v}(G) \\
&= \mathrm{LCErr}^{(m+1)}_{u\to v}(G) - \mathrm{LCErr}^{(0)}_{u\to v}(G).
\end{aligned}$$

Since $\mathrm{LCErr}^{(0)}_{u\to v}(G) = 0$, we are done. (If $j \geq k$, then $\mathrm{LCErr}_{u\to v}(G^{(m)}) = \mathrm{LCErr}^{(m+1)}_{u\to v}(G) = 0$, so the lemma holds trivially in this case.) □

The following corollary corresponds to Eq. (3).

Corollary 3.4. *If G fools every subprogram $f_{u\to v}$ with error α, then for every $m \geq 0$, $G^{(m)}$ fools f with error $wn \cdot (2w^2 n\alpha)^{m+1}$.*

Proof. For any u and v, by Lemma 3.1,

$$\begin{aligned}
|\,\mathrm{LCErr}_{u\to v}(G^{(m)})| &= |\,\mathrm{LCErr}^{(m+1)}_{u\to v}(G)| \\
&\leq \sum_{u=u_0,u_1,\ldots,u_{m+1}=v} \prod_{j=1}^{m+1} |\,\mathrm{LCErr}_{u_{i-1}\to u_i}(G)| \\
&\leq (wn)^m \cdot (2w\alpha)^{m+1} && \text{(Lemma 2.6)} \\
&\leq (2w^2 n\alpha)^{m+1}.
\end{aligned}$$

The corollary follows by Lemma 2.6. □

We reiterate that Corollary 3.4 follows already from the work of Cohen et al. and Pyne and Vadhan [CDRSTS21; PV21], and indeed the proof we have given is not substantially different (see Appendix A). In keeping with our remark after Lemma 2.6, we also remark that there is a version of Corollary 3.4 that eliminates the factors of w by assuming that for each layer j, the *sum* of errors $|G[u \to v] - U_n[u \to v]|$ over all $v \in V_j$ is at most α. We omit the details.

4 Our Improved WPRG for ROBPs

In this section, we will show how to convert a moderate-error PRG for width-w length-n ROBPs into a low-error WPRG for width-w length-n ROBPs. If the given PRG has error $1/\operatorname{poly}(wn)$ and seed length r, then for any $\varepsilon > 0$, we will obtain a WPRG with error ε and seed length $O(r + \log(wn/\varepsilon))$.

4.1 Construction

Recall the standard notion of an *averaging sampler*, which is essentially equivalent to a seeded randomness extractor [Zuc97].

Definition 4.1 (Sampler)**.** A function $\mathsf{Samp}\colon \{0,1\}^\ell \times \{0,1\}^q \to \{0,1\}^r$ is an (α, γ)-*sampler* if for every function $f\colon \{0,1\}^r \to [0,1]$,

$$\Pr_{x \in \{0,1\}^\ell}\left[\left|\mathbb{E}[f] - 2^{-q}\sum_{y \in \{0,1\}^q} f(\mathsf{Samp}(x,y))\right| \le \alpha\right] \ge 1 - \gamma.$$

Let $\alpha = \frac{1}{4w^3n^2}$ and let $G\colon \{0,1\}^r \to \{0,1\}^n$ be a given α-PRG for width-w length-n ROBPs. Define

$$m = \left\lceil \frac{\log(wn/\varepsilon)}{\log(wn)} \right\rceil \qquad \text{and} \qquad \gamma = \frac{\varepsilon}{2w^2n^2 \cdot ((8n)^{m+1}+1)} = \left(\frac{\varepsilon}{wn}\right)^{O(1)},$$

and let $\mathsf{Samp}\colon \{0,1\}^\ell \times \{0,1\}^q \to \{0,1\}^r$ be an (α,γ)-sampler. For each $x \in \{0,1\}^\ell$, let G_x be the distribution $G(\mathsf{Samp}(x, U_q))$, and let $G_x^{(m)}$ be the corresponding pseudodistribution with amplified local consistency as defined in Section 3.1. Our final pseudodistribution G' is $G_x^{(m)}$ for a uniform random x, i.e.,

$$G' = \sum_{x \in \{0,1\}^\ell} 2^{-\ell} \cdot G_x^{(m)}.$$

4.2 Correctness

Claim 4.2. *If f is a width-w length-n ROBP, then G' fools f with error ε.*

Proof. For each pair of vertices u, v, since G is an α-PRG for width-w ROBPs, G fools $f_{u\to v}$ with error α. Therefore, by the sampler condition, with probability $1-\gamma$ over a uniform random choice of x, G_x fools $f_{u\to v}$ with error 2α. Let BAD be the set of x such that there exist vertices u, v such that G_x does *not* (2α)-fool $f_{u\to v}$. By the union bound,

$$|\mathsf{BAD}| \le \gamma \cdot w^2n^2 \cdot 2^\ell = \frac{\varepsilon}{2 \cdot ((8n)^{m+1}+1)} \cdot 2^\ell.$$

For any x, unpacking the definitions, we see that $G_x^{(m)}$ has the form $\sum_{i=1}^K \pm A_i$, where

$$K \le (m+1) \cdot (n+1) \cdot (n+1)^m \cdot 2^m \le (8n)^{m+1}$$

and each A_i is a tensor product of probability distributions. Therefore, for $x \in \mathsf{BAD}$ (indeed for all x), we have $\left|\widetilde{\mathbb{E}}[f(G_x^{(m)})]\right| \le (8n)^{m+1}$. Meanwhile, for $x \notin \mathsf{BAD}$, by Corollary 3.4,

$$\left|\widetilde{\mathbb{E}}[f(G_x^{(m)})] - \mathbb{E}[f]\right| \le wn \cdot (4w^2 n\alpha)^{m+1} = wn \cdot \left(\frac{1}{wn}\right)^{m+1} < \frac{\varepsilon}{2}.$$

Therefore, overall,

$$\begin{aligned}
\left|\widetilde{\mathbb{E}}[f(G')] - \mathbb{E}[f]\right| &= \left|\sum_{x\in\mathsf{BAD}} 2^{-\ell} \cdot (\widetilde{\mathbb{E}}[f(G_x^{(m)})] - \mathbb{E}[f]) + \sum_{x\notin\mathsf{BAD}} 2^{-\ell} \cdot (\widetilde{\mathbb{E}}[f(G_x^{(m)})] - \mathbb{E}[f])\right| \\
&\le \sum_{x\in\mathsf{BAD}} 2^{-\ell} \left(\left|\widetilde{\mathbb{E}}[f(G_x^{(m)})]\right| + 1\right) + \sum_{x\notin\mathsf{BAD}} 2^{-\ell} \cdot \left|\widetilde{\mathbb{E}}[f(G_x^{(m)})] - \mathbb{E}[f]\right| \\
&\le 2^{-\ell} \cdot |\mathsf{BAD}| \cdot ((8n)^{m+1} + 1) + \frac{\varepsilon}{2} \\
&\le \varepsilon.
\end{aligned}$$

□

4.3 Explicitness and Seed Length

We will instantiate Samp using the following explicit sampler.

Theorem 4.3 ([CL20, Appendix B])**.** *For every $r \in \mathbb{N}$ and every $\alpha, \gamma > 0$, there exists an (α, γ)-sampler $\mathsf{Samp}\colon \{0,1\}^\ell \times \{0,1\}^q \to \{0,1\}^r$ with $\ell = r + O(\log(1/\gamma) + \log(1/\alpha))$ and $q = O(\log(1/\alpha) + \log\log(1/\gamma))$, such that given r, α, γ, x, and y, the value $\mathsf{Samp}(x, y)$ can be computed in space $O(r + \log(1/\alpha) + \log(1/\gamma))$.*

Proof of Theorem 1.5. Taking Samp to be the sampler of Theorem 4.3, we get $\ell = O(r + \log(1/\gamma)) = O(r + \log(wn/\varepsilon))$ and $q = O(\log(wn) + \log\log(1/\varepsilon))$. For a fixed $x \in \{0,1\}^\ell$, as mentioned in the proof of Claim 4.2, $G_x^{(m)}$ has the form $\sum_{i=1}^K \pm A_i$, where

$$K \le (8n)^{m+1} \le \mathrm{poly}(n/\varepsilon),$$

and each A_i is a tensor product of at most $2m+1$ terms, each of which is either $(G_x)_d$ for some value of d or else U_1. Using the constructions of Definition 2.4, we can sample $G_x^{(m)}$ by a WPRG with seed length $O(\log K + mq)$, and we can sample G' by a WPRG with seed length

$$\ell + O(\log K + mq) = O\left(r + \log(wn/\varepsilon) \cdot \left(1 + \frac{\log\log(1/\varepsilon)}{\log(wn)}\right)\right) = O(r + \log(wn/\varepsilon)),$$

where the last equality holds without loss of generality, because either $\varepsilon > 2^{-n}$, in which case $\log\log(1/\varepsilon) < \log(wn)$, or else $\varepsilon \le 2^{-n}$, in which case we can achieve seed length $O(r + \log(wn/\varepsilon))$ by simply sampling a truly random n-bit string. Furthermore, as discussed in Definition 2.4, our WPRG is $(2K)$-bounded,[8] and we can assume without loss of generality that $\varepsilon < 1/n$ (since otherwise G itself would be a suitable WPRG), so our WPRG is indeed $\mathrm{poly}(1/\varepsilon)$-bounded.

Finally, pick G to be Nisan's generator [Nis92]. Then

$$r = O(\log(wn/\alpha)\log n) = O(\log(wn)\log n),$$

so our WPRG has seed length $O(\log(wn)\log n + \log(1/\varepsilon))$ as claimed. Explicitness is clear. □

[8] The factor of 2 is because the number of terms in the sum might not be a power of two, so we might need to pad with dummy terms.

We remark that because of the specific structure of Nisan's generator [Nis92], the sampler is actually not necessary. Instead, we can let x be the description of the hash functions in Nisan's generator and let y be the input to the hash functions.

5 Derandomization Beyond Saks-Zhou

In this section, we show that randomized space-S decision algorithms can be simulated deterministically in space $O(S^{3/2}/\sqrt{\log S})$ (Theorem 1.6). As outlined in Section 1.6.2, the proof does not involve any particularly creative new ideas, but rather amounts to combining several previous works and choosing parameters. For that reason, we will refrain from fully describing the Saks-Zhou method. Instead, we will focus on assisting readers who are already familiar with Saks and Zhou's work [SZ99] (but who are not necessarily comfortable with WPRGs) in verifying Theorem 1.6. Readers who are not familiar with Saks and Zhou's work are encouraged to read Chattopadhyay and Liao's discussion of the Saks-Zhou method in the context of WPRGs [CL20] or Saks and Zhou's original paper [SZ99].

Let G denote Nisan's PRG [Nis92]. Recall that Saks and Zhou [SZ99] exploited the fact that the seed of Nisan's PRG can be split into two parts (x, y), where $x = O(\log w \log n)$ and $y = O(\log w)$; for a fixed ROBP f, if we pick x at random, then with high probability, $\mathbb{E}[f] \approx 2^{-r} \cdot \sum_y f(G(x, y))$. This method of estimating $\mathbb{E}[f]$ has a key technical feature, which is that each time we read a bit of the input of f, we only need to be using $O(\log w)$ bits of work space (not counting the string x, which we think of as being on a read-only random tape). This feature is beneficial, because in the context of the Saks-Zhou algorithm [SZ99], the program f is computed recursively, so we would like to use as little space as possible while the recursive computation is taking place. (See the work of Hoza and Umans for further discussion [HU21].)

Armoni generalized Saks and Zhou's methods by showing that *any* explicit PRG for ROBPs implies a method of estimating $\mathbb{E}[f]$ with the same key feature [Arm98]. Later, Chattopadhyay and Liao generalized further by showing that the same holds for any polynomially-bounded explicit WPRG [CL20]. For clarity, we repeat the argument here (in a slightly different form). It is convenient to generalize the definition of ROBPs to allow a large alphabet.

Definition 5.1 (ROBP over a large alphabet)**.** A width-w length-n *ROBP over the alphabet* Σ is defined as in Definition 1.1, except that each vertex in V_{i-1} has $|\Sigma|$ outgoing edges leading to V_i, labeled with the symbols in Σ. The program computes a function $f\colon \Sigma^n \to \{0,1\}$ in the natural way.

Lemma 5.2 ([CL20])**.** *Let $n = n(w)$, $K = K(w)$, $r = r(w)$, $a = a(w)$, and $\varepsilon = \varepsilon(w)$ be functions, each of which can be constructed in space $O(r)$. Suppose that for every $w \in \mathbb{N}$, there is a K-bounded ε-WPRG (G, ρ) for width-w length-n ROBPs over the alphabet $\{0,1\}^a$ with seed length r that can be computed in space $O(r)$. Then there is an algorithm for estimating the expectation of a given width-w length-n ROBP f over the alphabet $\{0,1\}^a$ with the following properties.*

1. *The algorithm uses $r + O(\log(Kw/\varepsilon))$ random bits from a read-only two-way[9] random tape, and with probability $1 - \varepsilon/w^2$ it outputs a value that is within $\pm 2\varepsilon$ of $\mathbb{E}[f]$.*
2. *The algorithm uses $O(r + a + \log(Kwn/\varepsilon))$ bits of work space. Furthermore, whenever the algorithm reads a bit of the description of f, it first deletes all but $O(a + \log(Kwn/\varepsilon))$ bits of its work space.*

[9] I.e., the algorithm is allowed to go back and re-read random bits as many times as it likes, unlike the standard model of randomized space-bounded computation in which the random tape must be read a single time from left to right.

Proof. Let $\mathsf{Samp}\colon \{0,1\}^\ell \times \{0,1\}^q \to \{0,1\}^r$ be the $(\varepsilon/(2K), \varepsilon/w^2)$-sampler of Theorem 4.3. To estimate $\mathbb{E}[f]$, we pick $x \in \{0,1\}^\ell$ uniformly at random, and then we output

$$2^{-q} \cdot \sum_{y \in \{0,1\}^q} \rho(\mathsf{Samp}(x,y)) \cdot f(G(\mathsf{Samp}(x,y))).$$

To prove that this works, define $g\colon \{0,1\}^r \to [-K,K]$ by $g(z) = \rho(z) \cdot f(G(z))$. The sampler condition implies that with probability $1 - \varepsilon/w^2$ over the choice of x, we have

$$\left| \mathbb{E}[g] - 2^{-q} \sum_{y \in \{0,1\}^q} g(\mathsf{Samp}(x,y)) \right| \leq \varepsilon.$$

Meanwhile, the WPRG condition implies that $|\mathbb{E}[g] - \mathbb{E}[f]| \leq \varepsilon$. Thus, with probability $1 - \varepsilon/w^2$, our algorithm outputs $\mathbb{E}[f] \pm 2\varepsilon$.

Now let us analyze the efficiency of the algorithm. The number of random bits we use is clearly $\ell = r + O(\log(Kw/\varepsilon))$. The total space used is $O(r)$ bits to compute G and ρ, plus $O(r \log(Kw/\varepsilon))$ bits to compute Samp, plus $O(\log(wn))$ bits to keep track of our simulation of f, plus $O(q) = O(\log(K/\varepsilon) + \log\log w)$ bits for summing over all y, which is a total of $O(r + a + \log(Kwn/\varepsilon))$ bits. Prior to reading a bit of the description of f, we only need to be storing the $O(\log(wn))$ bits that keep track of our simulation of f, plus the $O(q) = O(\log(K/\varepsilon) + \log\log w)$ bits for summing over all y, plus a single a-bit symbol of the string $G(\mathsf{Samp}(x,y))$ (namely, the single symbol that we are currently feeding into our simulation of f), which is a total of $O(a + \log(wnK/\varepsilon))$ bits. □

Having established Lemma 5.2, we can now compute the space complexity of the derandomization obtained by plugging any efficient WPRG into the Saks-Zhou framework.

Theorem 5.3 ([SZ99; CL20])**.** *Let $n = n(w)$, $K = K(w)$, and $r = r(w)$ be monotone increasing functions, each of which can be constructed in space $O(r)$, with $n \leq w$. Define $\varepsilon = w^{-8}$ and $a = \lceil 4 \log w \rceil$. Suppose that for every $w \in \mathbb{N}$, there exists a K-bounded ε-WPRG for width-$(w+1)$ length-n ROBPs over the alphabet $\{0,1\}^a$ with seed length r that can be computed in space $O(r)$. Then*

$$\mathbf{BPL} \subseteq \bigcup_{c \in \mathbb{N}} \mathbf{DSPACE}\left(r(N^c) + \frac{\log(N \cdot K(N^c)) \cdot \log N}{\log(n(N^c))} \right),$$

where N denotes the input length.

Proof outline. Suppose we are interested in computing the n-th power of a given substochastic matrix $M \in \mathbb{R}^{w \times w}$, where each entry has a bits of precision. We can easily construct a width-$(w+1)$ length-n ROBP f over the alphabet $\{0,1\}^a$ such that for each $i, j \in [w]$, if we let u_i be the i-th vertex in the first layer of f and we let v_j be the j-th vertex in the final layer of f, then $\mathbb{E}[f_{u_i \to v_j}] = (M^n)_{i,j}$. Using Lemma 5.2, we can compute each such value $\mathbb{E}[f_{u_i \to v_j}]$ to within $\pm 2\varepsilon$ with failure probability ε/w^2. In this way, we compute a matrix $P \in \mathbb{R}^{w \times w}$ such that $\|P - M^n\|_{\max} \leq 2\varepsilon$. We can reuse the same random bits for each entry of the matrix, so our algorithm uses $r + O(\log(Kw/\varepsilon))$ random bits from a read-only two-way random tape and succeeds with probability $1 - \varepsilon$. Furthermore, this algorithm uses $O(r + a + \log(Kwn/\varepsilon))$ bits of work space, and whenever it reads a bit of the description of M, it first deletes all but $O(a + \log(Kwn/\varepsilon))$ bits of its workspace.

Now, consider some randomized log-space algorithm that we wish to derandomize. There is a constant c such that the acceptance probability of the **BPL** algorithm on an input of length N is an entry in M^w, where $w = N^c$ and $M \in \{0, \frac{1}{2}, 1\}^{w \times w}$ is an easily-computable stochastic matrix.

We have discussed a randomized algorithm for approximating M^n. The technique of Saks and Zhou [SZ99] implies [CL20] an algorithm for computing M^w. As a reminder, the approach is to repeatedly take approximate n-th powers, reusing the same random bits each time and randomly rounding each entry of the matrix to a bits of precision after each iteration to resolve dependency issues. The number of iterations is $\frac{\log w}{\log n}$. The algorithm can be implemented to have failure probability $O(w^3 \cdot (2^a\varepsilon + 2^{-a}))$ and approximation error $O(w^2 2^{-a})$, using

$$O\left(r + \log(Kw/\varepsilon) + a \cdot \frac{\log w}{\log n}\right)$$

random bits and

$$O\left(r + (a + \log(Kwn/\varepsilon)) \cdot \frac{\log w}{\log n}\right)$$

bits of space [CL20, Lemma 43]. By our choices $\varepsilon = w^{-8}$ and $a = \lceil 4 \log w \rceil$, the failure probability is $O(1/w)$, the approximation error is $O(1/w^2)$, the number of random bits is $O(r + \log(Kw) + \frac{\log^2 w}{\log n})$, and the space complexity is $O(r + \frac{\log(Kw)\log w}{\log n})$. Trying all possibilities for the random tape completes the proof. □

Next, we identify the WPRG family that we will plug into Theorem 5.3.

Theorem 5.4 ([Arm98; KNW08; CDRSTS21])**.** *For every $w \in \mathbb{N}$, there exists a K-bounded ε-WPRG for width-$(w+1)$ length-n ROBPs over the alphabet $\{0,1\}^{\lceil 4 \log w \rceil}$ with seed length r that can be computed in space $O(r)$, where*

$$n = \exp\left(\left\lceil \sqrt{\log w \cdot \log\log w} \right\rceil\right), \qquad \varepsilon = w^{-8},$$
$$r \le O\left(\frac{\log^{3/2} w}{\sqrt{\log\log w}}\right), \qquad K \le \mathrm{poly}(w).$$

Proof. For any w, n, a, α, Armoni designed an α-PRG for width-$(w+1)$ length-n ROBPs over the alphabet $\{0,1\}^a$ [Arm98]; with an optimization due to Kane, Nelson, and Woodruff [KNW08], this PRG has seed length

$$r = O\left(a + \frac{\log(wn/\alpha)\log n}{\max\{1, \log\log w - \log\log(n/\alpha)\}}\right)$$

and can be computed in space $O(r)$. For $n = \exp\left(\left\lceil \sqrt{\log w \cdot \log\log w} \right\rceil\right)$, $\alpha = 1/\mathrm{poly}(n)$, and $a = O(\log w)$, this seed length becomes

$$r = O\left(\frac{\log^{3/2} w}{\sqrt{\log\log w}}\right).$$

Now we apply an error reduction procedure that converts this moderate-error PRG into a low-error WPRG. Specifically, we will use the reduction due to Cohen, Doron, Renard, Sberlo, and Ta-Shma [CDRSTS21]. Given a PRG for width-w length-n ROBPs over the alphabet $\{0,1\}^a$ with error $1/(10n^2)$ and seed length r, they show [CDRSTS21, Corollary 15] how to construct a WPRG for width-w length-n ROBPs over the alphabet $\{0,1\}^a$ with any desired error ε and seed length $r + O(\log(w/\varepsilon)\log\log_n(1/\varepsilon))$. Furthermore, if the PRG can be computed in space $O(r)$, then the WPRG can be computed in space $O(r + \log\log_n(1/\varepsilon) \cdot (\log\log(w/\varepsilon))^2)$. Cohen et al. did not explicitly mention it, but by inspection it is easy to see that their WPRG is $\mathrm{poly}(1/\varepsilon)$-bounded for the same reason that our main WPRG (Theorem 1.5) is $\mathrm{poly}(1/\varepsilon)$-bounded. Since $\varepsilon = 1/\mathrm{poly}(w)$, the seed length is $r + \widetilde{O}(\log w) = O(r)$, the space complexity is $O(r + \mathrm{poly}(\log\log w)) = O(r)$, and the WPRG is $\mathrm{poly}(w)$-bounded. □

Corollary 5.5. $\mathbf{BPL} \subseteq \mathbf{DSPACE}(\log^{3/2} N/\sqrt{\log\log N})$, *where* N *denotes the input length.*

Proof. Plugging the WPRG of Theorem 5.4 into Theorem 5.3, we get a space bound of

$$O\left(\frac{\log^{3/2}(N^c)}{\sqrt{\log\log(N^c)}} + \frac{\log(N \cdot N^{O(c)}) \cdot \log N}{\sqrt{\log(N^c) \cdot \log\log(N^c)}}\right),$$

which simplifies to $O(\log^{3/2} N/\sqrt{\log\log N})$. □

Finally, Theorem 1.6 follows from Corollary 5.5 by a standard padding argument.

6 Directions for Further Research

As we mentioned in Section 1.4, getting a better WPRG for width-n length-n ROBPs requires beating Nisan's PRG in the standard constant-error regime. However, there are cases where focusing on error dependence might still be fruitful:

- Recall that Nisan and Zuckerman designed a PRG for width-w length-n ROBPs when $n = \operatorname{polylog} w$ with optimal seed length $O(\log w)$ [NZ96] but non-optimal error $2^{-\log^{0.99} w}$. There are known ε-HSGs for this setting with seed length $O(\log w)$ even when $\varepsilon = 1/\operatorname{poly}(w)$ [AKS87; HZ20]; can we match that seed length by a WPRG? The WPRG construction presented in this paper does not seem to work, because if G has seed length $o(\log w)$, then it seems to have too much error for the local consistency amplification procedure $G^{(m)}$ to work, whereas if G has seed length $\Omega(\log w)$, then we cannot afford to sample multiple independent seeds.

- Currently, the best explicit PRGs for width-3 ROBPs and constant-width regular ROBPs have seed length $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$ [MRT19; De11; BRRY14]. In a similar spirit as Pyne and Vadhan's recent work on permutation ROBPs [PV21], can we design WPRGs for these models with error $1/\operatorname{poly}(n)$ and seed length $o(\log^2 n)$?

We also wonder whether there are other applications of recent WPRG constructions. For example, recall that Nisan showed $\mathbf{BPL} \subseteq \mathbf{DTISP}(\operatorname{poly}(n), \log^2 n)$ [Nis94]. Can we somehow use WPRGs to simulate **BPL** by a deterministic algorithm that simultaneously uses $\operatorname{poly}(n)$ time and $o(\log^2 n)$ space?

7 Acknowledgments

I thank David Zuckerman for helpful comments on a draft of this paper. I thank Alicia Hoza for suggesting ways to cut down on footnotes.

References

[AKMPSV20] A. Ahmadinejad, J. Kelner, J. Murtagh, J. Peebles, A. Sidford, and S. Vadhan. "High-precision Estimation of Random Walks in Small Space". In: *Proceedings of the 61st Annual Symposium on Foundations of Computer Science (FOCS)*. 2020, pp. 1295–1306. DOI: 10.1109/FOCS46700.2020.00123.

[AKS87] M. Ajtai, J. Komlos, and E. Szemeredi. "Deterministic Simulation in LOGSPACE". In: *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*. New York, New York, USA: Association for Computing Machinery, 1987, 132–140. ISBN: 0897912217. DOI: 10.1145/28395.28410. URL: https://doi.org/10.1145/28395.28410.

[Arm98] Roy Armoni. "On the derandomization of space-bounded computations". In: *Randomization and Approximation Techniques in Computer Science (RANDOM)*. Vol. 1518. Lecture Notes in Comput. Sci. Springer, Berlin, 1998, pp. 47–59. DOI: 10.1007/3-540-49543-6_5. URL: https://doi.org/10.1007/3-540-49543-6_5.

[BCG20] Mark Braverman, Gil Cohen, and Sumegha Garg. "Pseudorandom pseudo-distributions with near-optimal error for read-once branching programs". In: *SIAM J. Comput.* 49.5 (2020), STOC18–242–STOC18–299. ISSN: 0097-5397. DOI: 10.1137/18M1197734. URL: https://doi.org/10.1137/18M1197734.

[BCP83] A. Borodin, S. Cook, and N. Pippenger. "Parallel computation for well-endowed rings and space-bounded probabilistic machines". In: *Inform. and Control* 58.1-3 (1983), pp. 113–136. ISSN: 0019-9958. DOI: 10.1016/S0019-9958(83)80060-6. URL: https://doi.org/10.1016/S0019-9958(83)80060-6.

[BNS92] László Babai, Noam Nisant, and Márió Szegedy. "Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs". In: *Journal of Computer and System Sciences* 45.2 (1992), pp. 204–232. ISSN: 0022-0000. DOI: https://doi.org/10.1016/0022-0000(92)90047-M. URL: https://www.sciencedirect.com/science/article/pii/002200009290047M.

[BRRY14] Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. "Pseudorandom generators for regular branching programs". In: *SIAM J. Comput.* 43.3 (2014), pp. 973–986. ISSN: 0097-5397. DOI: 10.1137/120875673. URL: https://doi.org/10.1137/120875673.

[CDRSTS21] Gil Cohen, Dean Doron, Oren Renard, Ori Sberlo, and Amnon Ta-Shma. *Error Reduction For Weighted PRGs Against Read Once Branching Programs*. 2021. ECCC: TR21-020.

[CH20] Kuan Cheng and William M. Hoza. "Hitting Sets Give Two-Sided Derandomization of Small Space". In: *Proceedings of the 35th Annual Computational Complexity Conference (CCC)*. Ed. by Shubhangi Saraf. Vol. 169. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, 10:1–10:25. ISBN: 978-3-95977-156-6. DOI: 10.4230/LIPIcs.CCC.2020.10. URL: https://drops.dagstuhl.de/opus/volltexte/2020/12562.

[CL20] Eshan Chattopadhyay and Jyun-Jie Liao. "Optimal Error Pseudodistributions for Read-Once Branching Programs". In: *Proceedings of the 35th Annual Computational Complexity Conference (CCC)*. Ed. by Shubhangi Saraf. Vol. 169. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, 25:1–25:27. ISBN: 978-3-95977-156-6. DOI: 10.4230/LIPIcs.CCC.2020.25. URL: https://drops.dagstuhl.de/opus/volltexte/2020/12577.

[De11] Anindya De. "Pseudorandomness for permutation and regular branching programs". In: *Proceedings of the 26th Annual Conference on Computational Complexity (CCC)*. IEEE Computer Soc., Los Alamitos, CA, 2011, pp. 221–231.

[HPV21] William M. Hoza, Edward Pyne, and Salil Vadhan. "Pseudorandom Generators for Unbounded-Width Permutation Branching Programs". In: *Proceedings of the 12th Annual Innovations in Theoretical Computer Science Conference (ITCS)*. Ed. by James R. Lee. Vol. 185. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2021, 7:1–7:20. ISBN: 978-3-95977-177-1. DOI: 10.4230/LIPIcs.ITCS.2021.7. URL: https://drops.dagstuhl.de/opus/volltexte/2021/13546.

[HU21] William M. Hoza and Chris Umans. "Targeted Pseudorandom Generators, Simulation Advice Generators, and Derandomizing Logspace". In: *SIAM Journal on Computing* (2021), STOC17–281–STOC17–304. DOI: 10.1137/17M1145707. eprint: https://doi.org/10.1137/17M1145707. URL: https://doi.org/10.1137/17M1145707.

[HZ20] William M. Hoza and David Zuckerman. "Simple optimal hitting sets for small-success **RL**". In: *SIAM J. Comput.* 49.4 (2020), pp. 811–820. ISSN: 0097-5397. DOI: 10.1137/19M1268707. URL: https://doi.org/10.1137/19M1268707.

[INW94] Russell Impagliazzo, Noam Nisan, and Avi Wigderson. "Pseudorandomness for Network Algorithms". In: *Proceedings of the 26th Annual Symposium on Theory of Computing (STOC)*. Montreal, Quebec, Canada: Association for Computing Machinery, 1994, 356–364. ISBN: 0897916638. DOI: 10.1145/195058.195190. URL: https://doi.org/10.1145/195058.195190.

[Jun81] H. Jung. "Relationships between probabilistic and deterministic tape complexity". In: *Proceedings of the 10th Annual Symposium on Mathematical Foundations of Computer Science (MFCS)*. Vol. 118. Lecture Notes in Comput. Sci. Springer, Berlin-New York, 1981, pp. 339–346.

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

[MRT19] Raghu Meka, Omer Reingold, and Avishay Tal. "Pseudorandom generators for width-3 branching programs". In: *Proceedings of the 51st Annual Symposium on Theory of Computing (STOC)*. ACM, New York, 2019, pp. 626–637. DOI: 10.1145/3313276.3316319. URL: https://doi.org/10.1145/3313276.3316319.

[Nis92] Noam Nisan. "Pseudorandom generators for space-bounded computation". In: *Combinatorica* 12.4 (1992), pp. 449–461. ISSN: 0209-9683. DOI: 10.1007/BF01305237. URL: https://doi.org/10.1007/BF01305237.

[Nis94] Noam Nisan. "RL $\subseteq$ SC". In: *Comput. Complexity* 4.1 (1994), pp. 1–11. ISSN: 1016-3328. DOI: 10.1007/BF01205052. URL: https://doi.org/10.1007/BF01205052.

[NZ96] Noam Nisan and David Zuckerman. "Randomness is linear in space". In: *J. Comput. System Sci.* 52.1 (1996), pp. 43–52. ISSN: 0022-0000. DOI: 10.1006/jcss.1996.0004. URL: https://doi.org/10.1006/jcss.1996.0004.

[PV21] Edward Pyne and Salil Vadhan. *Pseudodistributions That Beat All Pseudorandom Generators*. 2021. ECCC: TR21-019.

[Sav70] Walter J. Savitch. "Relationships between nondeterministic and deterministic tape complexities". In: *J. Comput. System Sci.* 4 (1970), pp. 177–192. ISSN: 0022-0000. DOI: 10.1016/S0022-0000(70)80006-X. URL: https://doi.org/10.1016/S0022-0000(70)80006-X.

[SZ99] Michael Saks and Shiyu Zhou. "$\mathrm{BP_HSPACE}(S) \subseteq \mathrm{DSPACE}(S^{3/2})$". In: *J. Comput. System Sci.* 58.2 (1999), pp. 376–403. ISSN: 0022-0000. DOI: 10.1006/jcss.1998.1616. URL: https://doi.org/10.1006/jcss.1998.1616.

[Zuc97] David Zuckerman. "Randomness-Optimal Oblivious Sampling". In: *Random Struct. Algorithms* 11.4 (Dec. 1997), 345–367. ISSN: 1042-9832.

A Local Consistency vs. Approximate Inverse Laplacian

Cohen et al. noted that their WPRG construction is reminiscent of local consistency errors [CDRSTS21]. We now briefly elaborate on the connection, for the sake of readers who are familiar with how prior work used preconditioned Richardson iteration to decrease error in space-bounded derandomization [AKMPSV20; CDRSTS21; PV21].

Prior works [AKMPSV20; CDRSTS21; PV21] looked at the transition probability matrix W associated with a width-w length-n ROBP f, considered as a directed graph on $(n+1) \cdot w$ vertices. This matrix W is an $(n+1)w \times (n+1)w$ block matrix of the form

$$W = \begin{bmatrix} 0 & M_1 & 0 & \cdots & 0 \\ 0 & 0 & M_2 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \ddots & M_n \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix},$$

where $M_i \in \{0, 1/2, 1\}^{w \times w}$ is the transition probability matrix for $V_{i-1} \times V_i$. Let $L = I - W$ (the Laplacian matrix). Then L is invertible with inverse

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

where $M_{i\ldots j} = M_i \cdot M_{i+1} \cdots M_j$, i.e., $M_{i\ldots j}$ is the stochastic matrix containing the probabilities $U_n[u \to v]$ for $u \in V_i$ and $v \in V_j$. We are interested in obtaining an approximation $\widehat{L^{-1}}$ to L, say

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

where each $\widehat{M_{i\ldots j}}$ is a matrix of estimates for the probabilities $U_n[u \to v]$ with $u \in V_i$ and $v \in V_j$. The approach taken by prior work [AKMPSV20; CDRSTS21; PV21] is to use preconditioned Richardson iteration to convert a moderate-error approximation of L^{-1} into a low-error approximation of L^{-1}.

The crucial point is that in this analysis, the approximation quality is measured *by comparing* $\widehat{L^{-1}}L$ *to* I rather than comparing L^{-1} and $\widehat{L^{-1}}$ directly. The error matrix $E \stackrel{\text{def}}{=} I - \widehat{L^{-1}}L$ is given

by

$$E = \begin{bmatrix} 0 & E_{0\ldots1} & E_{0\ldots2} & \cdots & E_{0\ldots n} \\ 0 & 0 & E_{1\ldots2} & \cdots & E_{1\ldots n} \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \ddots & E_{n-1\ldots n} \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix},$$

where

$$E_{i\ldots j} = \widehat{M_{i\ldots j-1}} M_j - \widehat{M_{i\ldots j}}.$$

Thus, E is precisely the matrix of local consistency errors.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il