2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)

2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS) | 979-8-3503-1894-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/FOCS57990.2023.00072

Weighted Pseudorandom Generators via Inverse Analysis of Random Walks and Shortcutting

Lijie Chen
Miller Institute for Basic Research in Science
University of California, Berkeley
Berkeley, US
wjmzmbr@gmail.com

William M. Hoza
Department of Computer Science
University of Chicago
Chicago, US
williamhoza@uchicago.edu

Xin Lyu
EECS
University of California, Berkeley
Berkeley, US
xinlyu@berkeley.edu

Avishay Tal
EECS
University of California, Berkeley
Berkeley, US
atal@berkeley.edu

Hongxun Wu
EECS
University of California, Berkeley
Berkeley, US
wuhx@berkeley.edu

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

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

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

***Index Terms*—Pseudorandomness, Branching Program**

Lijie Chen is supported by Miller Research Fellowship. Avishay Tal is supported by Sloan Research Fellowship and NSF CAREER Award CCF-2145474. Xin Lyu and Hongxun Wu are supported by Avishay Tal's Sloan Research Fellowship, NSF CAREER Award CCF-2145474, and Jelani Nelson's ONR grant N00014-18-1-2562. This work was done while William M. Hoza was visiting the Simons Institute for the Theory of Computing.

I. Introduction

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

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

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

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

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

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

Definition I.2 (Standard-order ROBPs)**.** *Let $w, n \in \mathbb{N}$. A width-w length-n* standard-order ROBP *is a directed graph B. The vertex set consists of $n+1$* layers, *$V = V^{(0)} \cup V^{(1)} \cup \cdots \cup V^{(n)}$, where $|V^{(i)}| \leq w$ for each i. We usually assume without loss of generality that $|V^{(i)}| = w$. Every vertex $v \in V^{(i)}$ with $i < n$ has two outgoing edges, one labeled 0 and the other labeled 1, leading to $V^{(i+1)}$. There is a designated* start vertex *$v_{\text{start}} \in V^{(0)}$, and there is a set of designated* accepting vertices *$V_{\text{accept}} \subseteq V^{(n)}$. The program computes a Boolean function $B\colon \{0,1\}^n \to \{0,1\}$ as follows. Given an input $x \in \{0,1\}^n$, let $v_{\text{start}} = v_0, v_1, v_2, \ldots, v_n$ be the unique path such that for each $i \in [n]$, there is an edge from v_{i-1} to v_i with label x_i. If $v_n \in V_{\text{accept}}$, then we set $B(x) = 1$, and otherwise $B(x) = 0$.*

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

Using the probabilistic method, one can show the existence of non-explicit ε-PRGs for width-w length-n ROBPs with seed length $O(\log(wn/\varepsilon))$. Furthermore, several unconditional constructions of explicit PRGs for standard-order ROBPs are known. Most famously, Nisan designed a PRG that ε-fools width-w length-n ROBPs with seed length $O(\log(wn/\varepsilon) \cdot \log n)$ [1]. Nisan's PRG has found numerous applications, but its seed length is too large to resolve the L vs. BPL problem.

A. Weighted PRGs

Nisan's PRG [1] is more than three decades old. Despite much effort, there is still no known explicit PRG for standard-order ROBPs of polynomial width (or even width 4) with a better seed length. This motivates the search for *alternative approaches* to proving $\mathsf{L} = \mathsf{BPL}$ that do not necessarily require any breakthroughs on the PRG problem.[2] Braverman, Cohen, and Garg introduced one such approach [3], based on the concept of a *weighted PRG* (WPRG).

Definition I.3 (WPRGs)**.** *Let $n \in \mathbb{N}$, let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$, and let $\varepsilon > 0$. An* ε-WPRG *for $\mathcal{F}$ is a pair $(\mathcal{G}, \mu)$, where $\mathcal{G}\colon \{0,1\}^s \to \{0,1\}^n$ and $\mu\colon \{0,1\}^s \to \mathbb{R}$, such that for every $f \in \mathcal{F}$, we have*

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

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

Crucially, the weights $\mu(u)$ are allowed to be negative. (Indeed, WPRGs with nonnegative weights are essentially equivalent to unweighted PRGs [4, Appendix C].) Intuitively, this means that we are considering the expectation of f with respect to a sparse input "distribution" in which *some probabilities are negative*. For this reason, WPRGs are also known as *pseudorandom pseudodistribution generators* [3].

Because we are allowed to use negative weights, constructing WPRGs is potentially easier than constructing unweighted PRGs. Indeed, Braverman, Cohen, and Garg [3] constructed an explicit WPRG that ε-fools width-w length-n standard-order ROBPs with seed length

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

which is better than Nisan's PRG's seed length when the error parameter ε is very small. A sequence of followup works developed simpler and better WPRG constructions [4]–[7], in particular improving the seed length to $O(\log(wn) \cdot \log n + \log(1/\varepsilon))$ [7]. These examples demonstrate that negative weights open up new avenues for making progress. Furthermore, for a certain class of branching programs, Pyne and Vadhan constructed a WPRG [4] with a seed length that is *provably impossible* to achieve via unweighted PRGs [8], demonstrating the *intrinsic* power of negative weights. (Jumping ahead, we improve Pyne and Vadhan's construction in this work. See Section I-D.)

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

Definition I.4 (HSGs)**.** *Let $n \in \mathbb{N}$, let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$, and let $\varepsilon > 0$. An* ε-HSG *for $\mathcal{F}$ is a function $\mathcal{G}\colon \{0,1\}^s \to \{0,1\}^n$ such that for every $f \in \mathcal{F}$, if $\mathbb{E}[f] > \varepsilon$, then there exists $u \in \{0,1\}^s$ such that $f(\mathcal{G}(u)) = 1$.*

If $(\mathcal{G}, \mu)$ is an ε-WPRG for $\mathcal{F}$, then $\mathcal{G}$ is an ε-HSG for $\mathcal{F}$ [3]. HSGs have been studied for many decades, perhaps starting with the pioneering work of Ajtai, Komlós, and Szemerédi [9]. It turns out that an explicit HSG for width-n length-n standard-order ROBPs with optimal

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

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

seed length $O(\log n)$ would already imply $\mathsf{L} = \mathsf{BPL}$ [10], [11], just like a PRG or a WPRG. However, in the non-optimal regime (which is the most relevant regime given the present state of knowledge), the known applications of WPRGs exceed the known applications of HSGs. In particular, the current state-of-the-art unconditional derandomization of space-bounded computation says that randomized space-S algorithms can be simulated deterministically in space $O(S^{3/2}/\sqrt{\log S})$ [7]. This result is a recent slight improvement over Saks and Zhou's decades-old $O(S^{3/2})$ bound [12]. The improvement relies on the recent line of work on WPRGs [3]–[7], and it is not clear how to reproduce the result using HSGs alone.

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

B. *Width-*3 *Branching Programs*

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

For width-2 programs, explicit PRGs are known with optimal seed length $O(\log(n/\varepsilon))$ [13]–[15]. The width-3 case is at the frontier of current knowledge. Meka, Reingold, and Tal designed an explicit ε-PRG for width-3 standard-order ROBPs with seed length $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$ [16]. When the error parameter ε is constant, Meka, Reingold, and Tal's seed length is near optimal, but when $\varepsilon = 1/\mathrm{poly}(n)$, their PRG does not beat Nisan's $O(\log^2 n)$ seed length. To address this issue, we present a WPRG that fools width-3 standard-order ROBPs with error $1/\mathrm{poly}(n)$ and seed length $\widetilde{O}(\log^{3/2} n)$. More generally, we achieve the following parameters.

Theorem I.5 (WPRG for width-3 ROBPs)**.** *For every $n \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit ε-WPRG with seed length*

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

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

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

[3]Note that there is also work on width-3 ROBPs in the more challenging arbitrary-order setting [16], [18].

C. *Regular Branching Programs*

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

Definition I.6 (Regular ROBPs)**.** *Let B be a standard-order ROBP with vertex set $V = V^{(0)} \cup \cdots \cup V^{(n)}$. We say that B is* regular *if every vertex $v \in V \setminus V^{(0)}$ has precisely two incoming edges.*

Regular ROBPs have been the subject of intense study [19]–[24]. One reason to study regular ROBPs is that there is a reduction from the general case to the regular case. Indeed, Lee, Pyne, and Vadhan recently showed that if a function $f\colon \{0,1\}^n \to \{0,1\}$ can be computed by a standard-order ROBP of width w, then it can also be computed by a standard-order *regular* ROBP of width $O(wn)$ [25]. (This is an improvement over previous reductions [19], [23].) Consequently, optimal explicit WPRGs for polynomial-width standard-order *regular* ROBPs would imply optimal explicit WPRGs for polynomial-width standard-order *non-regular* ROBPs.

Braverman, Rao, Raz, and Yehudayoff designed an explicit ε-PRG for width-w length-n standard-order regular ROBPs with seed length $\widetilde{O}(\log(w/\varepsilon) \cdot \log n)$ [20]. (See also De's followup work [21].) When the width parameter w and the error parameter ε are both constant, Braverman, Rao, Raz, and Yehudayoff's seed length is near-optimal. However, when $\varepsilon = 1/\mathrm{poly}(n)$, their seed length is no better than Nisan's $O(\log^2 n)$ seed length, even if we hold w constant. To address this issue, we present a WPRG that fools constant-width standard-order regular ROBPs with error $1/\mathrm{poly}(n)$ and seed length $\widetilde{O}(\log^{3/2} n)$. More generally, our WPRG has the following parameters.

Theorem I.7 (WPRG for regular ROBPs)**.** *For every $n, w \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit ε-WPRG with seed length*

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

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

Our seed length matches that of an HSG for the same class that Bogdanov, Hoza, Prakriya, and Pyne recently constructed [23] (ignoring $\log\log$ factors). In turn, their construction is the best HSG known for this class (again, ignoring $\log\log$ factors). See Table II.

[4]Note that there is also work on the intriguing setting of unbounded-width programs with only one accepting vertex, as well as the challenging arbitrary-order setting [23], [26]. Furthermore, Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan gave a near-optimal non-black-box algorithm for estimating the acceptance probability of a given standard-order regular ROBP [27]; see Section I-E.

Seed length	Type	Reference	Notes
$\widetilde{O}(\log(n/\varepsilon))$	HSG	[17]	
$\widetilde{O}(\log n \cdot \log(1/\varepsilon))$	PRG	[16]	
$\widetilde{O}\left(\log n \cdot \sqrt{\log(1/\varepsilon)} + \log(1/\varepsilon)\right)$	WPRG	This work	
$O(\log(n/\varepsilon))$	PRG	Folklore	Optimal; non-explicit

TABLE I
PSEUDORANDOMNESS CONSTRUCTIONS FOR WIDTH-3 STANDARD-ORDER ROBPS.[3]

Seed length	Type	Reference	Notes
$\widetilde{O}(\log(w/\varepsilon) \cdot \log n)$	PRG	[20]	
$O(w \cdot \log n)$ for all $\varepsilon > 0$	HSG	[20]	
$\widetilde{O}\left(\log n \cdot \left(\log w + \sqrt{\log(1/\varepsilon)}\right) + \log(1/\varepsilon)\right)$	HSG	[23]	
$\widetilde{O}\left(\log n \cdot \left(\log w + \sqrt{\log(1/\varepsilon)}\right) + \log(1/\varepsilon)\right)$	WPRG	This work	
$O(\log(wn/\varepsilon))$	PRG	Folklore	Optimal; non-explicit

TABLE II
PSEUDORANDOMNESS CONSTRUCTIONS FOR WIDTH-w LENGTH-n STANDARD-ORDER REGULAR ROBPS.[4]

D. Unbounded-Width Permutation Branching Programs

Definition I.8 (Permutation ROBPs). *Let B be a standard-order ROBP with vertex set $V = V^{(0)} \cup \cdots \cup V^{(n)}$. We say that B is a* permutation ROBP *if every vertex $v \in V \setminus V^{(0)}$ has precisely two incoming edges, and those two incoming edges have distinct labels.*

In other words, between every two adjacent layers of a permutation ROBP, the edges labeled 0 form a matching, as do the edges labeled 1. Permutation ROBPs are thus a subclass of regular ROBPs. The first sequence of papers studying permutation ROBPs focused on the constant-width regime [21], [22], [28]–[31]. In recent years, there has been another wave of interest in permutation ROBPs, but this time, the focus is on programs of *unbounded width* with only one accepting vertex [4], [8], [23], [24], [32], [33]. This intriguing model cannot be considered a model of "space-bounded" computation anymore; instead, it is a certain type of "reversible" computation. Still, we hope that studying this model can shed light on the L vs. BPL problem. Studying unbounded-width models has already been a fruitful approach for proving new results about standard bounded-width models [4], [24], [33]. Indeed, in general, results for unbounded-width models typically imply corresponding results for standard width-w models, with a factor-of-w loss in the error parameter. Additionally, the unbounded-width model serves as a valuable "test-bed" for studying different notions of pseudorandomness.

Hoza, Pyne, and Vadhan designed an ε-PRG for these programs with seed length $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$ [8], building heavily on prior work by Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan [27]. Importantly, Hoza, Pyne, and Vadhan also proved a near-matching seed length *lower bound*: every ε-PRG for this model has seed length at least $\widetilde{\Omega}(\log n \cdot \log(1/\varepsilon))$ [8]. In contrast, Pyne and Vadhan designed an ε-WPRG for this model with seed length $\widetilde{O}(\log n \cdot \sqrt{\log(n/\varepsilon)} + \log(1/\varepsilon))$ [4]. In particular, when $\varepsilon = 1/\mathrm{poly}(n)$, the optimal seed length for unweighted PRGs is $\Theta(\log^2 n)$, whereas Pyne and Vadhan's WPRG seed length is only $\widetilde{O}(\log^{3/2} n)$. As mentioned previously, these results demonstrate that WPRGs have an intrinsic advantage over unweighted PRGs in this case.

A weakness of Pyne and Vadhan's WPRG [4] is that the seed length is always at least $\log^{3/2} n$. We present a new WPRG that smoothly interpolates between Hoza, Pyne, and Vadhan's $\widetilde{O}(\log n)$ seed length in the constant-error regime [8] and Pyne and Vadhan's $\widetilde{O}(\log^{3/2} n)$ seed length in the inverse-polynomial-error regime [4].

Theorem I.9 (WPRG for unbounded-width permutation ROBPs). *For every $n \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit WPRG with seed length*

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

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

Thus, our work strengthens the known separation between WPRGs and unweighted PRGs. Note that our seed length is always at least as good as Hoza, Pyne, and Vadhan's PRG seed length [8] (ignoring log log factors). See Table III for a summary. We also remark that as a simple corollary of Theorem I.9, we obtain an explicit WPRG for width-w standard-order permutation ROBPs (with an unbounded number of accepting vertices) with seed length

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

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

In addition to WPRGs, we also study non-black-box derandomization algorithms. In this model, we are given the complete description of an ROBP, and our job is to estimate its acceptance probability. In a remarkable recent work [27], Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan designed an algorithm for estimating the acceptance probability of a given regular ROBP within an inverse polynomial error using $\widetilde{O}(\log(wn))$ bits of space. Formally:

Theorem I.10 (High-precision non-black-box derandomization of regular ROBPs [27])**.** *There is a deterministic algorithm that uses $\widetilde{O}(\log(wn) \cdot \log\log(1/\varepsilon))$ bits of space and outputs a value that is within $\pm\varepsilon$ of $\mathbb{E}[B]$, given a width-w length-n standard-order regular ROBP B and a value $\varepsilon \in (0,1)$ as inputs.*

Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's work [27] combines techniques from the pseudorandomness literature with sophisticated machinery developed in a sequence of works on faster solvers for Eulerian directed Laplacian systems [34]–[36]. Roughly speaking, it introduced three new ideas to the space-bounded derandomization community: (1) the applicability of *Richardson iteration* for error reduction in the space-bounded setting, (2) the notion of the *unit-circle approximation*, and (3) a sophisticated analysis of the INW generator [37] via bounding a matrix norm defined by a recursive application of Schur complement (see [27, Theorem 6.1]).

The first two ideas have led to a flurry of exciting new results in space-bounded derandomization [4], [6]–[8], [38], [39], including much of the prior work on WPRGs that we described earlier. In contrast, the third idea has found limited applications so far, despite the fact that it is arguably the core of Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's proof [27]. Only one paper, by Pyne and Vadhan [4], has managed to adapt this technique to a new problem.

A possible explanation might be that it is not easy to understand the *meaning* of the constructed matrix norm [27, Section 6] in the context of derandomizing ROBPs. Moreover, Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's analysis [27] makes heavy use of complicated linear algebra facts developed in the Laplacian solver literature, and it is unclear how to interpret these facts when applied to the matrix constructed from a branching program.

To remedy this situation, we present a significantly simpler proof of Theorem I.10. In our proof, we view the recursive Schur complement construction in the original proof as a natural combinatorial "shortcut graph;" see Section III-A3 for details. We believe that this proof helps to clarify what exactly the recursive-Schur-complement-matrix-norm really represents in the setting of branching programs. We hope that our proof enables a fuller appreciation of Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's analysis [27] and thereby facilitates the development of new ideas.[7]

[6]Note that there is also work on the more challenging arbitrary-order setting [23], [24], [26]. Note also that the optimal WPRG seed length for this model is unclear.

[7]Indeed, all of our new WPRG constructions are inspired by the insights from our simpler proof of Theorem I.10.

II. Preliminaries

A. Read-Once Branching Programs

Let B be a standard-order ROBP (Definition I.2) with vertex set $V = V^{(0)} \cup \cdots \cup V^{(n)}$. Let $v_{\text{start}} \in V^{(0)}$ be the start vertex, and let $V_{\text{accept}} \subseteq V^{(n)}$ be the set of accepting vertices.

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

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

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

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

For convenience, we think of $B^{S\leftarrow u}$ as a program of length n rather than length $r-\ell$. That is, in $B^{S\leftarrow u}$, the first ℓ transitions are trivial identity layers; the next $r-\ell$ transitions are the same as in B; and the final $n-r$ transitions are trivial identity layers. Thus, we think of $B^{S\leftarrow u}$ as a Boolean function on n bits, but it only depends on $r-\ell$ of those bits:

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

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

The matrix notation $\mathbf{B}(x)$.: As explained in Definition I.2, we identify B with a Boolean function $B\colon \{0,1\}^n \to \{0,1\}$. We use boldface $\mathbf{B}$ to denote the matrix-valued function $\mathbf{B}\colon \{0,1\}^n \to \{0,1\}^{V^{(n)}\times V^{(0)}} \cong \{0,1\}^{w\times w}$ given by

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

Seed length	Type	Reference	Notes
$O(\log(n/\varepsilon)\cdot\log n)$	PRG	[21]	
$\widetilde{O}(\log n\cdot\log(1/\varepsilon))$	PRG	[8]	
$\widetilde{O}\left(\log n\cdot\sqrt{\log(n/\varepsilon)}+\log(1/\varepsilon)\right)$	WPRG	[4]	
$\widetilde{O}\left(\log n\cdot\sqrt{\log(1/\varepsilon)}+\log(1/\varepsilon)\right)$	WPRG	This work	
$\widetilde{\Omega}(\log n\cdot\log(1/\varepsilon))$	PRG	[8]	Lower bound
$O(\log(n/\varepsilon))$	HSG	[8]	Optimal; non-explicit
$O(\log(n/\varepsilon))$	Det. sampler[5]	[32]	Optimal; non-explicit

TABLE III

PSEUDORANDOMNESS CONSTRUCTIONS FOR UNBOUNDED-WIDTH PERMUTATION ROBPS WITH A SINGLE ACCEPTING STATE.[6]

ROBPs over large alphabets.: We occasionally use the standard large-alphabet generalization of Definition I.2. A width-w length-n standard-order ROBP *over the alphabet* Σ is defined just like Definition I.2, except that each vertex in $V^{(0)}\cup\cdots\cup V^{(n-1)}$ has $|\Sigma|$ outgoing edges leading to the next layer, labeled with the symbols in Σ. Thus, the program computes a function $B\colon\Sigma^n\to\{0,1\}$. We say that the program is *regular* if each vertex in $V^{(1)}\cup\cdots\cup V^{(n)}$ has precisely $|\Sigma|$ incoming edges, and we say that the program is a *permutation program* if those $|\Sigma|$ incoming edges have distinct labels.

B. Linear Algebra

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

Definition II.1 (Vector seminorm induced by a psd matrix)**.** *Let $\mathbf{M}\in\mathbb{R}^{N\times N}$ be a positive semidefinite matrix. We define a corresponding function $\|\cdot\|_{\mathbf{M}}\colon\mathbb{R}^N\to[0,\infty)$ by the rule*

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

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

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

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

Definition II.2 (Vector seminorm)**.** *A* seminorm *on $\mathbb{R}^N$ is a function $\|\cdot\|\colon\mathbb{R}^N\to[0,\infty)$ satisfying the following properties:*

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

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

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

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

Definition II.3 (Extended submultiplicative matrix seminorm)**.** *An* extended submultiplicative matrix seminorm *on $\mathbb{R}^{N\times N}$ is a function $\|\cdot\|\colon\mathbb{R}^{N\times N}\to[0,\infty]$ satisfying the following conditions.*

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

Definition II.4 (Extended submultiplicative matrix seminorm induced by a vector seminorm)**.** *Let $\|\cdot\|\colon\mathbb{R}^N\to[0,\infty)$ be a vector seminorm. We define an extended submultiplicative matrix seminorm $\|\cdot\|\colon\mathbb{R}^{N\times N}\to[0,\infty]$ by the rule*

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

III. TECHNICAL OVERVIEW

A. Our Error Reduction Framework

Each of our constructions follows the same high-level approach: We start with a construction that has moderate error $\tau\gg\varepsilon$, and then we apply an *error reduction* procedure to decrease the error down to ε. For example, for regular ROBPs, our WPRG construction (Theorem I.7) works by applying an error reduction procedure to the BRRY PRG [20]. Many recent works on space-bounded derandomization have developed and applied error reduction procedures [4], [6], [7], [23], [27], [38]–[40]. Like most of this prior work, our approach for error reduction is based on the "inverse Laplacian" perspective on space-bounded derandomization, introduced by Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan [27]. We review this perspective next.

1) The Inverse Laplacian Perspective: Let B be a width-w length-n standard-order ROBP. The vertices of B are denoted as $V(B) := V^{(0)} \cup V^{(1)} \cup \cdots \cup V^{(n)}$ where $V^{(i)} = \{i \cdot w + 1, \ldots, (i+1) \cdot w\}$ for each $i \in \{0, 1, \ldots, n\}$. Next, we define $\mathbf{W} \in \mathbb{R}^{(n+1) \cdot w \times (n+1) \cdot w}$ to be the transition matrix of B. Specifically, for every directed edge (u, v) in B, we set $\mathbf{W}_{v,u} = \frac{1}{2}$ (or $\mathbf{W}_{v,u} = 1$ if both outgoing edges of u go to v). [8] Since B only contains edges between pairs of adjacent layers, $\mathbf{W}$ has the following form:

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

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

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

Being a unitriangular matrix, $\mathbf{L}$ is invertible. Observe that (1) (See Figure 1) holds.

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

The inverse Laplacian perspective is easiest to understand in the non-black-box setting: if we are given the *description* of the ROBP B, then we can readily compute the Laplacian matrix $\mathbf{L}$, and it makes sense to try to approximately invert it. It turns out that the perspective is also valuable in the black-box setting. To construct WPRGs, our approach will be to first reason in terms of matrix arithmetic, and then "reverse engineer" a WPRG such that all the matrix arithmetic happens in the *analysis* of the WPRG rather than its construction. This technique is due to Cohen, Doron, Renard, Sberlo, and Ta-Shma [6] and, independently, Pyne and Vadhan [4]. For this technical overview, we primarily focus on the matrix arithmetic itself.

2) Richardson Iteration: With the inverse Laplacian perspective in mind, our high-level plan for proving our results is as follows. First, we will construct a matrix $\widehat{\mathbf{L}}^{-1}$ that approximates $\mathbf{L}^{-1}$ with moderate error.[9] Specifically, we will ensure that $\|\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L}\| \leq \delta$ for a suitable, sub-multiplicative matrix norm[10] $\|\cdot\|$ and some "moderately small" $\delta < 1$. Then, we will apply a powerful, generic error reduction technique called *Richardson iteration*: for each $m \in \mathbb{N}$, define

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

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

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

So, how shall we construct this matrix $\widehat{\mathbf{L}}^{-1}$? For each $w \times w$ block $\mathbf{W}_{j \leftarrow i}$ of $\mathbf{L}^{-1}$, we begin by constructing a matrix $\widetilde{\mathbf{W}}_{j \leftarrow i}$ that approximates $\mathbf{W}_{j \leftarrow i}$ with moderate error. For example, in the case of bounded-width standard-order regular ROBPs, we let $\widetilde{\mathbf{W}}_{j \leftarrow i}$ consist of the estimated probabilities of going from vertices in layer i to vertices in layer j based on the BRRY PRG [20], which gives us bounds such as $\|\widetilde{\mathbf{W}}_{j \leftarrow i} - \mathbf{W}_{j \leftarrow i}\|_1 \leq \tau$ where τ is moderately small.

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

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

Indeed, this approach has been used in prior work [6], [7]. The trouble with this approach is that if we defined $\widehat{\mathbf{L}}^{-1}$ in this way, then the error of the matrix $\widehat{\mathbf{L}}^{-1}$ as a whole would be typically much larger than the error of a single block. For example, the condition $\|\widetilde{\mathbf{W}}_{j \leftarrow i} - \mathbf{W}_{j \leftarrow i}\|_1 \leq \tau$ would merely give us $\|\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L}\|_1 \leq O(n \cdot \tau)$, essentially due to a "union bound" over the $n+1$ rows/columns. Therefore, if we used this approach, we would have to start with a very small initial error $\tau < \frac{1}{n}$ to get a nontrivial bound ($\|\mathbf{I} - \widehat{\mathbf{L}}^{-1}\mathbf{L}\| < 1$). In the

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

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

[10]Technically, in some cases we will use an "extended seminorm" rather than a true norm; see Definition II.3.

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

Fig. 1. Inverse of $\mathbf{L}$.

settings we study, we cannot afford to compute $\widetilde{\mathbf{W}}_{j\leftarrow i}$ with such a low error. For example, notice that all the explicit PRGs in Table I, Table II, and Table III have seed length $\Omega(\log n \cdot \log(1/\tau))$ for error τ. If used these PRGs with an initial error value of $\tau < 1/n$, then our seed lengths would always be $\Omega(\log^2 n)$. We therefore need a different approach.

3) Constructing $\widehat{\mathbf{L}}^{-1}$ via Shortcutting and Correction Graphs: Our construction of $\widehat{\mathbf{L}}^{-1}$ uses ideas from Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's algorithm for estimating random walk probabilities in Eulerian digraphs [27]. To briefly summarize their algorithm, they recursively applied Rozenman and Vadhan's derandomized square operation [41] to construct a "unit-circle approximation" with moderate error $\tau \approx 1/\log n$, and then they decrease the error using heavy machinery from the literature on fast Laplacian solvers.

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

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

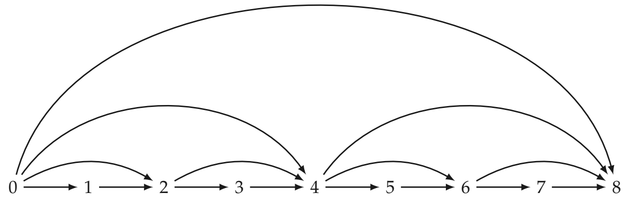

Fig. 2. The shortcut graph SC_n with $n = 8$.

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

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

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

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

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

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

Comparisons: Our construction of $\widehat{\mathbf{L}}^{-1}$ based on the shortcut graph corresponds to Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's recursive usage of the Schur complement operation [27, Theorem 6.1].

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

Analyzing $\widehat{\mathbf{L}}^{-1}$. Inverse analysis of random walks: Having defined $\widehat{\mathbf{L}}^{-1}$, our job is to bound $\|\mathbf{I}-\widehat{\mathbf{L}}^{-1}\mathbf{L}\|$, i.e., our job is to show that $\widehat{\mathbf{L}}^{-1}$ approximately describes random walks in the given branching program B in a certain sense. At an intuitive level, our approach is as follows. We admit that $\widehat{\mathbf{L}}^{-1}$ doesn't *perfectly* describe random walks in B; we only aim to show that it is an approximation. However, $\widehat{\mathbf{L}}^{-1}$ *does* perfectly describe random walks in *some other* graph $\widehat{B}$! This graph $\widehat{B}$ is not a true ROBP (it is not layered, and its edges have positive and negative weights), but still, our construction of $\widehat{\mathbf{L}}^{-1}$ ensures that $\widehat{B}$ has considerable combinatorial structure. We use this structure to show that $\widehat{B} \approx B$ in some sense, which enables us to bound the error $\|\mathbf{I}-\widehat{\mathbf{L}}^{-1}\mathbf{L}\|$. The process of going from $\widehat{\mathbf{L}}^{-1}$ to $\widehat{B}$ (i.e., starting with a description of the behavior of random walks, and reconstructing an underlying graph that is consistent with that description) is what we refer to as the "inverse analysis of random walks." [11]

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

$$\begin{aligned}&\text{Suppose } \|\mathbf{L}^{-1}\cdot(\widehat{\mathbf{L}}-\mathbf{L})\| \le \delta.\\ &\text{Then } \|\widehat{\mathbf{L}}^{-1}\cdot(\widehat{\mathbf{L}}-\mathbf{L})\| \le \delta/(1-\delta).\end{aligned} \tag{4}$$

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

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

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

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

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

For example, the inverse of the $n=4$ example from (3) is given in Figure 3: Since $\widehat{\mathbf{L}} = \mathbf{I}-(\mathbf{W}+\Delta\mathbf{W})$, the matrix $\widehat{\mathbf{L}}$ can be interpreted as the "Laplacian" of a weighted graph with transition matrix $\mathbf{W}+\Delta\mathbf{W}$. We refer to the subgraph corresponding to $\Delta\mathbf{W}$ as the "Correction Graph," because adding $\Delta\mathbf{W}$ to $\widehat{\mathbf{L}}$ yields the original Laplacian $\mathbf{L}$.

Overall, we have

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

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

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

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

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

B. Our WPRG for Bounded-Width Regular ROBPs

In this section, we explain how we carry out each step of the recipe to construct a WPRG that ε-fools bounded-width regular ROBPs (Theorem I.7). For ease of exposition, we will focus on the case $\varepsilon = 1/n$ in this proof overview, and we will assume that the width of the program is $O(1)$.

We construct the approximation matrices $\widetilde{\mathbf{W}}_{j\leftarrow i}$ by using the BRRY PRG [20]. We set the error parameter of the PRG to $\tau = 2^{-\sqrt{\log n}}$, so the BRRY PRG has seed length $\widetilde{O}(\log n\cdot\log(1/\tau)) = \widetilde{O}(\log^{3/2} n)$. We use the standard ℓ_∞ matrix norm, i.e.,

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

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

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

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

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

Fig. 3. Inverse of (3) when $n = 4$.

We analyze the matrix above using the *weight* measure introduced by Braverman, Rao, Raz, and Yehudayoff [20]. Using techniques similar to Braverman, Rao, Raz, and Yehudayoff's arguments [20], one can show that for any vertex u in layer ℓ and any vertex v in layer j, we have

$$|(\mathbf{W}_{j\leftarrow \ell+2} \cdot (\widetilde{\mathbf{W}}_{\ell+2\leftarrow \ell} - \mathbf{W}_{\ell+2\leftarrow \ell}))_{v,u}| \\ \leq O(\tau \cdot \mathsf{Weight}(B^{v\leftarrow}, \ell, \ell+2)), \quad (5)$$

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

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

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

A similar argument bounds $\|\mathbf{L}^{-1}\Delta\mathbf{W}^{(t)}\|$ when $t > 1$. From here, our error reduction framework provides a matrix $\mathbf{A}_m$ such that $\|\mathbf{I} - \mathbf{A}_m\mathbf{L}\| \leq \Theta(1/n)$. By reverse-engineering the definition of $\mathbf{A}_m$, we construct a WPRG that fools B with error $1/n$ as desired. However, the seed length of this WPRG is too large, because each seed includes several independent seeds for the BRRY PRG. We therefore decrease the seed length by using the Impagliazzo-Nisan-Wigderson (INW) PRG [37] to generate *correlated* seeds for the BRRY PRG, similar to prior work [4], [6].

C. *Our WPRG for Width-3 ROBPs*

Next, we give an overview of our WPRG for width-3 ROBPs (Theorem I.5). Recall that Meka, Reingold, and Tal designed a PRG for width-3 length-n standard-order ROBPs with seed length $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$ [16]. We do not apply our error reduction framework to this PRG in a black-box way. Instead, to construct our WPRG, we revisit Meka, Reingold, and Tal's specific construction and analysis [16].

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

1) The restriction ρ *preserves the expectation* of B, i.e., $|\mathbb{E}_{\rho,\mathcal{U}}[B|_\rho(\mathcal{U})] - \mathbb{E}[B]| \leq \varepsilon$, where $\mathcal{U}$ is independent of ρ and uniform random.
2) The program B *simplifies* under ρ with high probability. Indeed, it "almost" becomes a permutation ROBP.

In more detail, with probability $1 - \varepsilon$, the restricted program $B|_\rho$ is approximated by a program $\widetilde{B}$ that satisfies the permutation condition (Definition I.8) in all but a few layers. In this case, Meka, Reingold, and Tal argue that the BRRY PRG [20] fools $B|_\rho$ [16]. Therefore, to fool the original program B, they apply the pseudorandom restriction ρ and then fill in the stars using the BRRY PRG. The overall seed length is dominated by the $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$ seed length of the BRRY PRG.

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

A quantitative improvement. Fewer colliding layers: In Meka, Reingold, and Tal's analysis, the number of "colliding layers" in $\widetilde{B}$ (i.e., layers that violate the permutation condition) is bounded by $\mathrm{poly}(1/\varepsilon) \cdot \log\log n$ [16]. We are most interested in the regime $\varepsilon = 1/\mathrm{poly}(n)$, in which their bound is trivial. To address this issue, we show that with probability $1 - \varepsilon$, the restricted program $B|_\rho$ is in fact approximated by a program with only $\mathrm{polylog}(1/\varepsilon) \cdot \log\log n$ many colliding layers (see Claim 7.35 in the full version).

A qualitative improvement. Bounding the weight of the restricted program: Building on our analysis of colliding layers, we show that with high probability, the restricted function $B|_\rho$ can be computed by a program with bounded *weight* (as defined by Braverman, Rao, Raz, and Yehudayoff [20]). Specifically, with probability $1 - \varepsilon$, the weight is at most $\mathrm{polylog}(n/\varepsilon)$; see Theorem 7.2 of the full version.

We consider this "simplification" statement in terms of weight to be cleaner and easier to understand, compared to Meka, Reingold, and Tal's analysis [16]. Furthermore, it turns out to be a crucial step in our WPRG analysis. To explain why, let $\mathcal{X}$ be a pseudorandom string sampled by the BRRY PRG [20]. As mentioned previously, Meka, Reingold, and Tal prove that $\mathcal{X}$ fools the restricted program $B|_\rho$ [16]. Let us recall their argument in more detail, so that we can see where it breaks down when we try to use our WPRG in place of $\mathcal{X}$.

The first step of the argument is to show that $\mathcal{X}$ fools programs with few colliding layers such as $\widetilde{B}$ [16]. Then, to bridge the gap between $\widetilde{B}$ and $B|_\rho$, the second step is to design an "error indicator" function E with the following properties:

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

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

$$\left|\mathbb{E}[B|_\rho] - \mathbb{E}\left[\widetilde{B}\right]\right| \leq \mathbb{E}[E] \leq \varepsilon$$
$$\text{and } \left|\mathbb{E}[B|_\rho(\mathcal{X})] - \mathbb{E}\left[\widetilde{B}(\mathcal{X})\right]\right| \leq \mathbb{E}[E(\mathcal{X})] \leq \varepsilon, \quad (6)$$

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

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

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

due to cancellations in the right-hand sum. Therefore, (6) no longer works after we introduce negative weights. (In general, WPRGs do not seem to be "compatible" with sandwiching arguments.)

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

In the analysis of our WPRG for regular ROBPs, the only place we use regularity is to bound the weight of the program (and its subprograms[12]). Therefore, since we show that $B|_\rho$ can be computed by a program with low weight, it follows that our WPRG fools $B|_\rho$. By combining with the pseudorandom restriction ρ, we get a WPRG that fools B itself. For more details, see Section 7 of the full version.

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

D. Our Simplified Derandomization of Polynomial-Width Regular ROBPs

We now present an overview of our simplified proof of Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's main result [27] (Theorem I.10). That is, given the description of a regular width-w length-n standard-order ROBP B, we will explain how to estimate $\mathbb{E}[B]$ to within a small additive error in near-logarithmic space. The most important case to keep in mind is $w = \text{poly}(n)$.

Our proof is largely inspired by Braverman, Rao, Raz, and Yehudayoff's work [20]. To explain the intuition, let us begin by revisiting the analysis of *constant-width* regular ROBPs that we summarized in Section III-B, so we can understand more deeply why it works. Afterward, we will explain how to adapt the intuition to the polynomial-width case.

1) Reflections on Braverman, Rao, Raz, and Yehudayoff's Techniques [20]: Let $V^{(0)}, \ldots, V^{(n)}$ be the layers of a constant-width regular program B. Fix some target vertex v, say $v \in V^{(j)}$. To show that $\|\mathbf{L}^{-1}\Delta\mathbf{W}^{(1)}\|_\infty \leq O(\tau)$, where τ is the error of each block $\widetilde{\mathbf{W}}_{r\leftarrow\ell}$, the main steps that we discussed in Section III-B are as follows.

1) First, we *bound the "local" error in terms of weight*. Let $u \in V^{(\ell)}$ where $\ell \leq j-2$. We show
$$|(\mathbf{W}_{j\leftarrow\ell+2} \cdot (\widetilde{\mathbf{W}}_{\ell+2\leftarrow\ell} - \mathbf{W}_{\ell+2\leftarrow\ell}))_{v,u}| \leq O(\tau \cdot \mathsf{Weight}(B^{v\leftarrow}, \ell, \ell+2)). \quad (7)$$
2) Then, we *bound the total weight of the program*. Braverman, Rao, Raz, and Yehudayoff showed that $\mathsf{Weight}(B^{v\leftarrow}, 0, n) \leq O(1)$ [20]. Consequently, the total error is $O(\tau)$.

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

The BRRY potential argument: Let S be the set of vertices $q \in V^{(\ell+2)}$ that are reachable from u. (See Figure 4.) Define

$$\text{Spread}(u) := \left(\max_{q \in S} \mathbb{E}[B^{v\leftarrow q}]\right) - \left(\min_{q \in S} \mathbb{E}[B^{v\leftarrow q}]\right),$$

and consider two extreme cases:

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

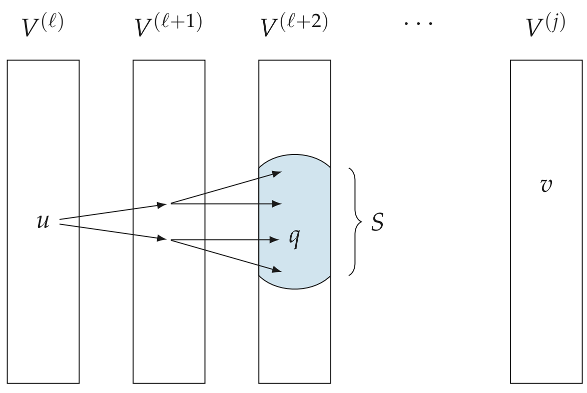

Fig. 4. The setup for the analysis of constant-width regular ROBPs.

walk, in which we reverse all the directions of the edges in B. The assumption $\text{Spread}(u) \gg 0$ means that there are two vertices in $V^{(\ell+2)}$ that have very different probabilities of being reached *from* v, and u is reachable from both of them. Intuitively, because B is regular, this should mean that in the reversed random walk, the distribution over states becomes *more uniform* (mixed) when we walk from $V^{(\ell+2)}$ to $V^{(\ell)}$.

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

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

which quantifies how far the reversed random walk is from uniform ("fully mixed") when it reaches $V^{(i)}$ [20]. For each $u \in V^{(\ell)}$, one can use the concept of "weight" to prove

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

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

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

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

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

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

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

Key new idea. Using the ℓ_2 norm: Why do we use the ℓ_∞ norm to measure $y_\perp^{(u)}$? Intuitively, the underlying reason is that we are working with an ℓ_∞-type guarantee regarding the approximation matrices $\widetilde{\mathbf{W}}_{j\leftarrow i}$. To get the best WPRG seed length, we construct $\widetilde{\mathbf{W}}_{j\leftarrow i}$ using the BRRY PRG [20] (as discussed in Section III-B), but the correctness proof works in the more general setting where $\widetilde{\mathbf{W}}_{j\leftarrow i}$ is constructed using an *arbitrary* PRG that fools constant-width regular ROBPs with moderate error. In such a general setting, using the ℓ_∞ norm seems necessary, because it captures the "worst possible error" for a pseudorandom walk starting from u.

To make progress in the polynomial-width case, we take a less "generic" approach. We construct the approximation matrices $\widetilde{\mathbf{W}}_{j\leftarrow i}$ by recursively applying (a version of) Rozenman and Vadhan's derandomized squaring operation [41] (similar to what Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan did [27]). This operation uses a *spectral expander graph* to recycle randomness. Let $\mathbf{H}$ be the expander's transition matrix, and let $\mathbf{J}$ be the transition matrix of a complete graph with self-loops. The quality of $\mathbf{H}$ is measured by the quantity $\lambda(\mathbf{H}) = \|\mathbf{H} - \mathbf{J}\|_2$, i.e., we measure error using the ℓ_2 norm. This raises the following question:

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

Although seemingly naïve, this idea is the core motivation for our simplified proof of Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's main result [27]. We believe it is the "real magic" behind their work.

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

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

$$\left|\beta^T \cdot (\mathbf{H} - \mathbf{J}) \cdot \alpha\right| \leq \lambda(\mathbf{H}) \cdot \|\alpha\|_2 \cdot \|\beta\|_2 \leq \frac{\lambda(\mathbf{H})}{2} \cdot (\|\alpha\|_2^2 + \|\beta\|_2^2). \quad (8)$$

Now let $x \in \mathbb{R}^{V^{(\ell)}}$ and $y \in \mathbb{R}^{V^{(\ell+2)}}$ be vectors. For each vertex $u \in V^{(\ell)}$, let $y^{(u)}$ be the subvector of y that is reachable from u. That is, if the i-th path from u leads to v, then we define $y^{(u)}_i = y_v$. Similarly, for each vertex $v \in V^{(\ell+2)}$, let $x^{(v)}$ be the subvector of x from which v is reachable. By regularity, $x^{(v)}$ and $y^{(u)}$ are both vectors in $\mathbb{R}^D$ where $D = 2^2 = 4$. Using (8), one can show that

$$\left| y^T \cdot \left(\widetilde{\mathbf{W}}_{\ell+2\leftarrow\ell} - \mathbf{W}_{\ell+2\leftarrow\ell} \right) \cdot x \right| \leq \frac{\lambda(\mathbf{H})}{2D} \cdot \left(\sum_{v \in V^{(\ell+2)}} \|x^{(v)}_{\perp}\|_2^2 + \sum_{u \in V^{(\ell)}} \|y^{(u)}_{\perp}\|_2^2 \right). \quad (9)$$

When x and y are chosen appropriately, the left-hand side of (9) measures the amount of error that occurs in layers ℓ through $\ell+2$. Therefore, provided that all of the ℓ_2 norms $\|x^{(v)}_{\perp}\|_2$ and $\|y^{(u)}_{\perp}\|_2$ are close to zero, (9) says that we are in an "error-free" case, as desired.

The sums on the right-hand side of (9) can be conveniently simplified as follows. Observe that $\|x^{(v)}_{\perp}\|_2^2 = \|x^{(v)}\|_2^2 - \|x^{(v)}_{\parallel}\|_2^2$. Intuitively, $x^{(v)}_{\parallel}$ corresponds to the v-th coordinate of $\mathbf{W}_{\ell+2\leftarrow\ell} \cdot x$. Using this idea, one can show that

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

and similarly,

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

Thus, (9) is equivalent to the statement that $\widetilde{\mathbf{W}}_{\ell+2\leftarrow\ell}$ is a good *singular-value approximation* of $\mathbf{W}_{\ell+2\leftarrow\ell}$, as defined by Ahmadinejad, Peebles, Pyne, Sidford, and Vadhan [42]:

Definition III.1 (Singular-value approximation [42]). *Let $\widetilde{\mathbf{W}}, \mathbf{W} \in \mathbb{R}^{w\times w}$ be doubly stochastic matrices. We say $\widetilde{\mathbf{W}}$ τ-singular-value approximates $\mathbf{W}$, denoted as $\widetilde{\mathbf{W}} \overset{\mathrm{sv}}{\approx}_{\tau} \mathbf{W}$, if for every $x, y \in \mathbb{R}^w$,*

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

In this proof overview, we are focusing on the length-two edges in SC_n, which correspond to a single application of the derandomized squaring operation. In the actual proof, we show that more generally, $\widetilde{\mathbf{W}}_{r\leftarrow\ell}$ is a good singular-value approximation of $\mathbf{W}_{r\leftarrow\ell}$ for every edge $(\ell, r) \in E(\mathsf{SC}_n)$. We would like to clarify that it was already known that recursive derandomized squaring produces a good singular-value approximation. Indeed, this fact readily follows from the analysis in Ahmadinejad, Peebles, Pyne, Sidford, and Vadhan's recent work [42].

3) The Mixing Case. Potential Dynamics: To summarize the discussion so far, our simplified proof of Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's main theorem [27] can be divided into two main parts.

1) First, we show that the approximation matrices $\widetilde{\mathbf{W}}_{j\leftarrow i}$ constructed via a version of the derandomized squaring operation [41] are singular-value approximations of the corresponding exact matrices $\mathbf{W}_{j\leftarrow i}$. (See Theorem 8.1 of the full version.)
2) Second, we show that $\|\mathbf{L}^{-1}\Delta\mathbf{W}\|$ is moderately small, where $\|\cdot\|$ is a suitably chosen matrix "norm."[13] (See Lemma 8.5 of the full version.)

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

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

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

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

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

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

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

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

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

The potential function: Analogous to the discussion in Section III-D1, our approach for bounding $\|z\|$ is to use a potential argument. Our potential function $\Phi\colon \mathbb{R}^w \to \mathbb{R}$ is given by

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

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

Observe that if y is a probability vector, then $\Phi(y)$ is a measure of how far y is from the uniform distribution. In this respect, $\Phi(\cdot)$ is similar to the potential function Φ_i discussed in Section III-D1; however, compared to Φ_i, our function $\Phi(\cdot)$ is more compatible with spectral analysis.

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

$$\begin{aligned}\underbrace{\left|\Phi(z^{[j+2]}) - \Phi(\widetilde{z}^{[j+2]})\right|}_{\text{amount of error in step } j} &\le O(\tau_0)\cdot\big(\underbrace{\Phi(z^{[j]}) - \Phi(\mathbf{W}_{j+2\leftarrow j}\cdot z^{[j]})}_{\text{mixing in } z} \\ &\quad + \underbrace{\Phi(x^{[j]}) - \Phi(\mathbf{W}_{j+2\leftarrow j}\cdot x^{[j]})}_{\text{mixing in } x}\big). \qquad (10)\end{aligned}$$

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

The $\mathbf{F}_1$*-seminorm.:* In light of (10), it is natural to define our "norm function" $\|\cdot\| : \mathbb{R}^{(n+1)\cdot w} \to \mathbb{R}$ by the formula

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

where $y = (y^{[0]},\ldots,y^{[n]})$. This quantity $\|y\|^2$ measures the *total potential drops* when we apply each matrix $\mathbf{W}_{j+2\leftarrow j}$ to the appropriate block of y. One can verify that this function $\|\cdot\|$ truly is a (semi)norm; we refer to it as the "$\mathbf{F}_1$-seminorm." Summing up (10) gives

$$\sum_{j\in\{0,2,4,\ldots,n\}} \underbrace{\left|\Phi(z^{[j+2]}) - \Phi(\widetilde{z}^{[j+2]})\right|}_{\text{amount of error in step } j} \le O(\tau)\cdot(\|z\|^2 + \|x\|^2). \qquad (11)$$

The left-hand side above is a measure of the total amount of error. However, our job is to bound $\|z\|$, which is, conceptually, a different way of measuring the total amount of error. To connect these two quantities, it might be helpful to visualize a dynamic process generating $z^{[0]}, \widetilde{z}^{[2]}, z^{[2]}, \ldots$ as in Figure 5.

The value $\|z\|^2$ is the sum of the heights of the dashed arrows in Figure 5. Meanwhile, the error measure on the left-hand side of (11) is the sum of the heights of the solid arrows in Figure 5. The key observation is that in this process, *the initial point $z^{[0]}$ is equal to the final point $z^{[n+2]}$* (both are the zero vector), so the two types of arrows must perfectly balance. Symbolically, we have

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

by (11). By choosing τ to be sufficiently small, we conclude that $\|z\| \le O(\sqrt{\tau}\cdot\|x\|)$ as desired.

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

E. Our WPRG for Unbounded-Width Permutation ROBPs

We conclude this technical overview by briefly discussing our improved WPRG for unbounded-width permutation ROBPs (Theorem I.9). This WPRG corresponds closely to our simplified proof of Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's main theorem [27]. We construct our WPRG by reverse-engineering the (non-black-box) algorithm for derandomizing regular ROBPs. In order to carry out this reverse-engineering process, we assume that the program we are fooling is a *permutation* program, because under this assumption, Rozenman and Vadhan's derandomized squaring operation [41] has a "black-box interpretation," namely, it is equivalent to the Impagliazzo-Nisan-Wigderson (INW) PRG [37]. The WPRG that we construct in this way has a seed length that is greater than what we can afford, because each seed includes several independent seeds for the INW PRG. We therefore decrease the seed length by using Hoza, Pyne, and Vadhan's PRG [8] to generate correlated seeds for the INW PRG.

The approach outlined above is essentially identical to the approach that Pyne and Vadhan use to construct their WPRG for permutation ROBPs [4]. The reason we get an improved bottom-line seed length is that Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan's algorithm [27] involves a "cycle lift" at some points. These cycle lifts are inherited by Pyne and Vadhan [4], and they effectively cost a factor of n in the

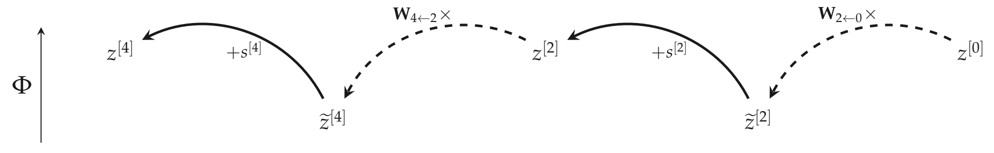

Fig. 5. An illustration of the dynamics of the potential function.

error parameter. In terms of seed length, this means that $O(\log n \cdot \sqrt{\log(1/\varepsilon)})$ becomes $O(\log^{3/2} n + \log n \cdot \sqrt{\log(1/\varepsilon)})$. In contrast, we never use any cycle lifts, hence we do not pay the extra $\log^{3/2} n$ term.

References

[1] N. Nisan, "Pseudorandom generators for space-bounded computation," *Combinatorica*, vol. 12, no. 4, pp. 449–461, 1992. [Online]. Available: https://doi.org/10.1007/BF01305237

[2] W. M. Hoza, "Recent progress on derandomizing space-bounded computation," *Bulletin of EATCS*, vol. 138, no. 3, 2022.

[3] M. Braverman, G. Cohen, and S. Garg, "Pseudorandom pseudo-distributions with near-optimal error for read-once branching programs," *SIAM J. Comput.*, vol. 49, no. 5, pp. STOC18–242–STOC18–299, 2020. [Online]. Available: https://doi.org/10.1137/18M1197734

[4] E. Pyne and S. P. Vadhan, "Pseudodistributions that beat all pseudorandom generators (extended abstract)," in *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, ser. LIPIcs, V. Kabanets, Ed., vol. 200. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 33:1–33:15. [Online]. Available: https://doi.org/10.4230/LIPIcs.CCC.2021.33

[5] E. Chattopadhyay and J.-J. Liao, "Optimal Error Pseudodistributions for Read-Once Branching Programs," in *35th Computational Complexity Conference (CCC 2020)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), S. Saraf, Ed., vol. 169. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, pp. 25:1–25:27. [Online]. Available: https://drops.dagstuhl.de/opus/volltexte/2020/12577

[6] G. Cohen, D. Doron, O. Renard, O. Sberlo, and A. Ta-Shma, "Error Reduction for Weighted PRGs Against Read Once Branching Programs," in *36th Computational Complexity Conference (CCC 2021)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), V. Kabanets, Ed., vol. 200. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, pp. 22:1–22:17.

[7] W. M. Hoza, "Better pseudodistributions and derandomization for space-bounded computation," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*, ser. LIPIcs, vol. 207. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 28:1–28:23.

[8] W. M. Hoza, E. Pyne, and S. Vadhan, "Pseudorandom generators for unbounded-width permutation branching programs," *12th Innovations in Theoretical Computer Science (ITCS'21)*, 2021.

[9] M. Ajtai, J. Komlós, and E. Szemerédi, "Deterministic simulation in logspace," in *Proceedings of the 19th Symposium on Theory of Computing (STOC)*, 1987, pp. 132–140.

[10] K. Cheng and W. M. Hoza, "Hitting sets give two-sided derandomization of small space," *Theory Comput.*, vol. 18, pp. Paper No. 21, 32, 2022. [Online]. Available: https://doi.org/10.4086/toc.2022.v018a021

[11] E. Pyne, R. Raz, and W. Zhan, "Certified hardness vs. randomness for log-space," ECCC preprint TR23-040, 2023. [Online]. Available: https://eccc.weizmann.ac.il/report/2023/040/

[12] M. E. Saks and S. Zhou, "BP $_{\rm h}$space(s) subseteq dspace(s$^{3/2}$)," *J. Comput. Syst. Sci.*, vol. 58, no. 2, pp. 376–403, 1999.

[13] M. Saks and D. Zuckerman, 1995, unpublished work showing that ε-biased distributions fool width-2 ROBPs with error $O(\varepsilon \cdot n)$.

[14] A. Bogdanov, Z. Dvir, E. Verbin, and A. Yehudayoff, "Pseudorandomness for width-2 branching programs," *Theory Comput.*, vol. 9, pp. 283–292, 2013. [Online]. Available: https://doi.org/10.4086/toc.2013.v009a007

[15] P. Hatami and W. M. Hoza, "Theory of unconditional pseudorandom generators," ECCC preprint TR23-019, 2023. [Online]. Available: https://eccc.weizmann.ac.il/report/2023/019/

[16] R. Meka, O. Reingold, and A. Tal, "Pseudorandom generators for width-3 branching programs," in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*. ACM, 2019, pp. 626–637.

[17] P. Gopalan, R. Meka, O. Reingold, L. Trevisan, and S. Vadhan, "Better pseudorandom generators from milder pseudorandom restrictions," in *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science—FOCS 2012*. IEEE Computer Soc., Los Alamitos, CA, 2012, pp. 120–129.

[18] T. Steinke, S. Vadhan, and A. Wan, "Pseudorandomness and Fourier-growth bounds for width-3 branching programs," *Theory Comput.*, vol. 13, pp. Paper No. 12, 50, 2017. [Online]. Available: https://doi.org/10.4086/toc.2017.v013a012

[19] O. Reingold, L. Trevisan, and S. Vadhan, "Pseudorandom walks on regular digraphs and the **RL** vs. **L** problem," in *STOC'06: Proceedings of the 38th Annual ACM Symposium on Theory of Computing*. ACM, New York, 2006, pp. 457–466. [Online]. Available: https://doi.org/10.1145/1132516.1132583

[20] M. Braverman, A. Rao, R. Raz, and A. Yehudayoff, "Pseudorandom generators for regular branching programs," *SIAM Journal on Computing*, vol. 43, no. 3, pp. 973–986, 2014.

[21] A. De, "Pseudorandomness for permutation and regular branching programs," in *26th Annual IEEE Conference on Computational Complexity*. IEEE Computer Soc., Los Alamitos, CA, 2011, pp. 221–231.

[22] O. Reingold, T. Steinke, and S. Vadhan, "Pseudorandomness for regular branching programs via Fourier analysis," in *Approximation, randomization, and combinatorial optimization*, ser. Lecture Notes in Comput. Sci. Springer, Heidelberg, 2013, vol. 8096, pp. 655–670. [Online]. Available: https://doi.org/10.1007/978-3-642-40328-6_45

[23] A. Bogdanov, W. M. Hoza, G. Prakriya, and E. Pyne, "Hitting sets for regular branching programs," in *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*, ser. LIPIcs, vol. 234. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, pp. 3:1–3:22.

[24] C. H. Lee, E. Pyne, and S. Vadhan, "Fourier growth of regular branching programs," in *Approximation, randomization, and combinatorial optimization. Algorithms and techniques*, ser. LIPIcs. Leibniz Int. Proc. Inform. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2022, vol. 245, pp. Art. No. 2, 21. [Online]. Available: https://doi.org/10.4230/lipics.approx/random.2022.2

[25] ——, "On the power of regular and permutation branching programs," *Electronic Colloquium on Computational Complexity (ECCC)*, 2023. [Online]. Available: https://eccc.weizmann.ac.il/report/2023/102/

[26] L. Chen, X. Lyu, A. Tal, and H. Wu, "New PRGs for Unbounded-Width/Adaptive-Order Read-Once Branching Programs," in *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), K. Etessami, U. Feige, and G. Puppis, Eds., vol. 261. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, pp. 39:1–39:20. [Online]. Available: https://drops.dagstuhl.de/opus/volltexte/2023/18091

[27] A. Ahmadinejad, J. A. Kelner, J. Murtagh, J. Peebles, A. Sidford, and S. P. Vadhan, "High-precision estimation of random walks in small space," in *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, S. Irani, Ed. IEEE, 2020, pp. 1295–1306. [Online]. Available: https://doi.org/10.1109/FOCS46700.2020.00123

[28] J. Brody and E. Verbin, "The coin problem and pseudorandomness for branching programs," in *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2010, pp. 30–39. [Online]. Available: https://www.cs.swarthmore.edu/~brody/papers/CoinProblemFullVersion.pdf

[29] M. Koucký, P. Nimbhorkar, and P. Pudlák, "Pseudorandom generators for group products [extended abstract]," in *STOC'11—Proceedings of the 43rd ACM Symposium on Theory of Computing*. ACM, New York, 2011, pp. 263–272. [Online]. Available: https://doi.org/10.1145/1993636.1993672

[30] T. Steinke, "Pseudorandomness for permutation branching programs without the group theory," ECCC preprint TR12-083, 2012. [Online]. Available: https://eccc.weizmann.ac.il/report/2012/083/

[31] E. Chattopadhyay, P. Hatami, K. Hosseini, and S. Lovett, "Pseudorandom generators from polarizing random walks," *Theory Comput.*, vol. 15, pp. Paper No. 10, 26, 2019. [Online]. Available: https://doi.org/10.4086/toc.2019.v015a010

[32] E. Pyne and S. Vadhan, "Deterministic approximation of random walks via queries in graphs of unbounded size," in *5th SIAM Symposium on Simplicity in Algorithms*. [Society for Industrial and Applied Mathematics (SIAM)], Philadelphia, PA, 2022, pp. 57–67.

[33] L. Golowich and S. Vadhan, "Pseudorandomness of expander random walks for symmetric functions and permutation branching programs," in *37th Computational Complexity Conference (CCC 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

[34] M. B. Cohen, J. A. Kelner, J. Peebles, R. Peng, A. Sidford, and A. Vladu, "Faster algorithms for computing the stationary distribution, simulating random walks, and more," in *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*. IEEE Computer Society, 2016, pp. 583–592.

[35] M. B. Cohen, J. A. Kelner, J. Peebles, R. Peng, A. B. Rao, A. Sidford, and A. Vladu, "Almost-linear-time algorithms for markov chains and new spectral primitives for directed graphs," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*. ACM, 2017, pp. 410–419. [Online]. Available: https://doi.org/10.1145/3055399.3055463

[36] M. B. Cohen, J. A. Kelner, R. Kyng, J. Peebles, R. Peng, A. B. Rao, and A. Sidford, "Solving directed laplacian systems in nearly-linear time through sparse LU factorizations," in *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*. IEEE Computer Society, 2018, pp. 898–909.

[37] R. Impagliazzo, N. Nisan, and A. Wigderson, "Pseudorandomness for network algorithms," in *STOC*. ACM, 1994, pp. 356–364.

[38] A. L. Putterman and E. Pyne, "Near-optimal derandomization of medium-width branching programs," *Electron. Colloquium Comput. Complex.*, vol. TR22-150, 2022.

[39] G. Cohen, D. Doron, O. Sberlo, and A. Ta-Shma, "Approximating iterated multiplication of stochastic matrices in small space," *Electron. Colloquium Comput. Complex.*, vol. TR22-149, 2022.

[40] W. M. Hoza and D. Zuckerman, "Simple optimal hitting sets for small-success **RL**," *SIAM J. Comput.*, vol. 49, no. 4, pp. 811–820, 2020. [Online]. Available: https://doi.org/10.1137/19M1268707

[41] E. Rozenman and S. Vadhan, "Derandomized squaring of graphs," in *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2005, pp. 436–447.

[42] A. Ahmadinejad, J. Peebles, E. Pyne, A. Sidford, and S. P. Vadhan, "Singular value approximation and reducing directed to undirected graph sparsification," *CoRR*, vol. abs/2301.13541, 2023.