Electronic Colloquium on Computational Complexity, Report No. 123 (2026)

A Forward–Backward Weight Analysis of INW for Permutation Branching Programs

Gil Cohen* Dean Doron† Noam Goldgraber‡

Abstract

We construct an ε-error PRG for permutation read-once branching programs of length n and width w with seed length

$$O\left(\left(\log w + \log(1/\varepsilon)\right) \cdot \log n\right).$$

This gives an exponential improvement in the dependence on w compared with the constructions of De (CCC 2011) and Steinke (ECCC 2012). Compared with the work of Braverman, Rao, Raz, and Yehudayoff (FOCS 2010; SICOMP 2014), which applies more generally to regular branching programs and already achieves the optimal dependence on w, our result improves the dependence on the length n, attaining the optimal logarithmic dependence.

The generator itself is the classical INW PRG of Impagliazzo, Nisan, and Wigderson (STOC 1994). We show that, for permutation branching programs, the INW generator can be instantiated with expanders whose degrees are polynomial in w and $1/\varepsilon$ and, crucially, independent of n. To prove this, we analyze error propagation using program-dependent seminorms tailored to the branching program at hand. These seminorms build on the weight function introduced by Braverman et al. The key point is that, when measured in these adapted seminorms, the error *does not accumulate* throughout the recursion.

Since our analysis relies only on the spectral expansion of the underlying expanders, our seed length tightly matches the recent lower bound for spectral analyses of the INW generator due to Hoza, Pyne, and Vadhan (Algorithmica 2024).

*Tel Aviv University. gil@tauex.tau.ac.il. Supported by ERC starting grant 949499 and by the Israel Science Foundation grant 2989/24.

†Ben Gurion University. deand@bgu.ac.il. Supported in part by NSF-BSF grant 2022644.

‡Ben Gurion University and Tel Aviv University. goldgrab@post.bgu.ac.il. Supported by NSF-BSF grant 2022644.

ISSN 1433-8092

Contents

1 Introduction

The **BPL** vs. **L** problem asks whether every probabilistic algorithm can be fully derandomized with only a constant factor blowup in space. The way that a space-bounded algorithm acts on its random bits can be modeled by a *read-once branching program* (BP, for short; see Definition 2.1). A natural and well-studied approach to derandomize **BPL** is to come up with a *pseudorandom generator* (PRG) that fools BPs with logarithmic seed length.

The seminal work of Nisan [Nis92] gave a PRG with seed length $O(\log^2 n)$ for BPs over n variables and width $w = \mathrm{poly}(n)$ – the setting that is required for derandomizing **BPL** via standard derandomization. Impagliazzo, Nisan, and Wigderson [INW94] subsequently introduced a second construction based on expander graphs. Although giving similar parameters to those in [Nis92], the INW PRG turned out to be more flexible, and will be the starting point for much of the discussion below.

Before we continue, let us briefly recall the structure of the INW generator. The construction is recursive: a generator for programs of length m is used to construct a generator for programs of length $2m$. The generator is applied to the first half of the program, and an expander graph is then used to "recycle" its seed into a correlated seed for the second half. Since the recursion has depth $\log n$, the final seed length is $O(\log d \cdot \log n)$, where d is the degree of the expanders. In the standard INW analysis, these expanders can be chosen to have degree d which is polynomial in n, w, and $1/\varepsilon$, yielding seed length

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

In particular, in the **BPL** regime, this yields seed length $O(\log^2 n)$.

Unfortunately, even for width $w = 4$, no construction is known with better than quadratic dependence on $\log n$. The only known exceptions are widths 2 and 3. Saks and Zuckerman showed that small-bias spaces fool width-2 BPs [SZ95] (see also [BDVY13]). The width-3 case, which is considerably more challenging, was treated by Meka, Reingold, and Tal [MRT19], who constructed a PRG with near-optimal dependence on n.

On the bright side, substantial progress has been made in constructing PRGs for more restricted classes of branching programs, a direction that has been studied extensively and successfully, with applications beyond the original **BPL** setting. A prominent example is the class of *permutation* BPs, which has received considerable attention. These are branching programs in which, at every layer, both the 0-transition and the 1-transition are permutations of the state space. Note that such a computation is reversible.

For this class, previous work achieved the optimal logarithmic dependence on the length n, and in fact, often using a substantially sharper analyses of the INW generator. Koucký, Nimbhorkar, and Pudlák [KNP11] analyzed the INW generator for group products, a structured

subclass of permutation branching programs. For general permutation BPs, their work achieved the optimal dependence on the length n, albeit with exponential dependence on the width w; more precisely, they obtained seed length

$$O((w! + \log(1/\varepsilon)) \cdot \log n).$$

De [De11] subsequently achieved polynomial dependence on w, obtaining seed length

$$O\big((w^8 + \log(1/\varepsilon)) \cdot \log n\big).$$

De's analysis uses representation-theoretic ideas. Steinke [Ste12] later gave a more elementary analysis, avoiding any reference to groups or representation theory, though still relying on rather intricate combinatorial arguments. This improved the width dependence to

$$O\big((w^4 \log w + \log(1/\varepsilon)) \cdot \log n\big).$$

This line of work shows that, for permutation BPs, the expanders underlying the INW generator can be chosen to have degree independent of the program length n, though exponential in the width w. Consequently, for constant width and constant error, the resulting seed length is $O(\log n)$.

A more general class of branching programs is that of *regular* BPs, in which every state has in-degree 2. Braverman, Rao, Raz, and Yehudayoff [BRRY14] gave a refined analysis of the INW generator for regular BPs, obtaining seed length

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

BRRY showed that, for regular BPs, it suffices to use expanders whose degree is polynomial only in the recursion depth, namely in $\log n$, rather than in the program's length n; this is the source of the $\log\log n$ term in their seed length. Their argument is substantially simpler than the analyses of De and Steinke [De11, Ste12], which achieve optimal dependence on n for permutation BPs, while also allowing the expander degree to depend only polynomially on w.

Nevertheless, the BRRY bound remains suboptimal in its dependence on the length. Within the INW framework, achieving seed length $O(\log n)$ requires expander degrees that are completely independent of n. Consequently, even a slowly growing loss across the recursion is prohibitive, and one must show that the relevant error does not accumulate at all.

1.1 Our Result

Despite a resurgence over the past five years in the study of PRGs and weighted PRGs [1] for permutation BPs and their unbounded-width variants, as discussed in Section 1.2, there had

[1] A weighted PRG is a relaxation of the standard notion of a PRG, introduced in [BCG20], in which each seed is assigned a real weight. These weights may be negative and need not be bounded in absolute value.

been no improvement for more than a decade on the original problem of constructing a PRG for permutation BPs, following the works of [De11, Ste12, BRRY14]. In this work, we obtain the first such improvement by giving a new analysis of the INW generator. We show that it fools permutation BPs with seed length $O\left((\log w + \log(1/\varepsilon)) \cdot \log n\right)$.

Compared with the analyses of De [De11] and Steinke [Ste12], which also achieve the optimal dependence on n, our result improves the dependence on w exponentially, attaining the optimal dependence on this parameter. Compared with the BRRY analysis [BRRY14], which applies more generally to regular BPs, our result achieves the optimal logarithmic dependence on the length n.

Theorem 1.1 (Main result; informal)**.** *There exists an explicit PRG $G : \{0,1\}^s \to \{0,1\}^n$ that ε-fools length-n width-w permutation branching programs, with seed length*

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

In particular, G is the INW generator, properly instantiated.

For the complete formal statement, see Theorem 3.1.

Because our proof is a spectral analysis, the resulting seed length tightly matches the lower bound for spectral analyses established by Hoza, Pyne, and Vadhan [HPV24], discussed below in Section 1.2. Thus, we answer affirmatively the first open problem posed in their work.

1.2 Other Related Work

As noted above, the study of PRGs and weighted PRGs for permutation BPs, restricted subclasses, and related variants has seen significant activity in recent years. In this section, we survey the results most relevant to our work.

For restricted families of permutation BPs, improved seed lengths are known via generators that are not based on the INW construction. In a paper predating the definition of permutation BPs, Lovett, Reingold, Trevisan, and Vadhan [LRTV09] constructed a PRG for the special case of abelian groups with seed length

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

More recently, for p-groups of constant size, Lee and Viola [LV25] obtained the optimal seed length $O(\log(n/\varepsilon))$ via a reduction to the construction of PRGs for polynomials over $\mathbb{F}_p$.

In a different line of work, more closely related to ours in its techniques, Pyne and Vadhan [PV21] constructed a weighted PRG for permutation BPs with seed length

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

Their construction improves upon the known PRG constructions in the low-error regime. Hoza, Pyne, and Vadhan [HPV21] constructed a PRG for unbounded-width permutation BPs with a single accepting state, having seed length

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

More recently, Cheng and Wu [CW26] obtained a weighted PRG for unbounded-width permutation BPs with seed length

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

Hoza, Pyne, and Vadhan [HPV24] studied the limitations of the INW generator and proved that any "spectral analysis" of it requires seed length

$$\Omega\left((\log w + \log(1/\varepsilon)) \cdot \log n\right),$$

even for permutation BPs. Here, a spectral analysis is one that assumes only a suitable spectral gap for the underlying expanders, without exploiting any additional properties. This framework encompasses all previous analyses of the INW generator. As noted above, our proof is spectral in this sense, and consequently, our result is tight within this framework.

Interestingly, in the more general setting of permutation BPs over an alphabet of size d, [HPV24] proved a lower bound of $\Omega(\log n \cdot \log\log \min\{n,d\})$. Our framework, however, relies crucially on the binary-alphabet setting. Standard reductions yield a PRG whose seed length has logarithmic dependence on d, and we leave closing the resulting gap as an open problem.

1.3 Proof Idea

A central ingredient in the BRRY analysis of the INW generator is the notion of the *weight* of a branching program. This quantity measures, in an averaged sense, how much the individual layers can influence the final value of the computation. BRRY used this notion to sharpen the error analysis of INW: instead of accumulating error linearly over the n layers of the program, the error is controlled in terms of the recursion depth, which is logarithmic in n, and the weight of the program. The second key step in their analysis is to prove that, for regular BPs, the total weight is bounded by $\mathrm{poly}(w)$ – independent of the length of the program. Combining these two ingredients yields their improved seed length.

Our key idea is to combine the BRRY weight perspective with the reversibility available in permutation BPs. In addition to the usual BRRY weight, which we view as a *backward* weight, we introduce a complementary *forward* weight corresponding to the reversed program. This symmetric use of forward and backward weights is the main new ingredient in our analysis.

It allows us to avoid paying for the recursive depth altogether, thereby yielding the optimal dependence on n.

Exploiting this symmetry requires the generator itself to preserve a compatible forward–backward structure. In our analysis, the seed-recycling step is not treated by an arbitrary sampler, but rather as the averaging operator of an expander, and can hence also recycle the seed when considering the inverse program. This feature is crucial for simultaneously controlling the forward and backward quantities that arise in the analysis. For this reason, our proof relies essentially on the expander-based structure of the INW construction. Interestingly, although for a seemingly different reason, a recent work by Hoza and Lv [HL25] also relies heavily on the fact that the INW generator is based on expander graphs.

At a conceptual level, our proof can also be viewed as an error-propagation analysis tailored to the branching program at hand. Rather than controlling the error throughout the recursion using a single fixed norm, such as the spectral norm or an induced ℓ_1 norm, as in most previous analyses, we associate with each subprogram its own seminorm, capturing the particular "geometry" of that subprogram. The main point is that, with respect to these adapted error measures, the error *does not accumulate* over the recursion: it remains bounded by $O(\lambda)$, where λ is the spectral expansion of the underlying expanders, provided λ is taken to be sufficiently small, namely inverse-polynomial in w and, crucially, independent of n.

While most prior work measures error using a fixed norm, several recent works instead consider program-dependent norms or seminorms [MRSV19, AKM+20, PV21, HPV21, APP+23, CHL+23] to give a better error reduction procedures—either algorithmically in a white-box setting or in the construction of weighted PRGs—or for controlling the error propagation in INW-inspired constructions throughout the recursion. We discuss the relationship between the seminorm used in these works and our seminorm, in greater detail in Section 4.1, focusing on [CHL+23] (that builds on the singular-value approximation [APP+23]).

Beyond its conceptual appeal, we believe that the seminorm formulation may be useful for combining our argument with other work. Many analyses of PRGs and related recursive constructions are phrased in terms of norm bounds, and the forward–backward seminorm isolates the precise norm-like quantity that our proof controls.

Organization. After the brief preliminaries in Section 2, we prove our main result, Theorem 1.1, in Section 3. The proof makes no explicit reference to the seminorms discussed above. We believe that presenting the argument in this form makes it more direct and easier to follow. In Section 4, we reinterpret the proof from the perspective of the forward–backward seminorms. Finally, in Section 5, we give a refined analysis of the proof of Theorem 1.1. This refinement may be useful in future work and also brings out a convolutional perspective on the argument.

2 Preliminaries

2.1 Branching Programs

We recall the relevant definitions of read-once branching programs, and PRGs that fool them. Those definitions are standard, and we refer the reader to [HH24] for a more detailed discussion.

Definition 2.1 (Branching programs (BPs))**.** *Let n, w be positive integers. A width-w length-n read-once branching program, denoted by P, is a directed graph on the set of vertices, called "states", which is arranged in layers $V = V_0 \cup \cdots \cup V_n$, each of size w. For every $t \in [n]$ and a vertex $v \in V_{t-1}$, there are two outgoing edges from v to V_t, with labels 0 and 1. There is a "start vertex" $s \in V_0$, and a set of "accept states" $A \subseteq V_n$. A string $x \in \{0,1\}^n$ defines a walk through the program – start at s, and at each step $t \in [n]$, walk through the outgoing edge which is labeled $x_t \in \{0,1\}$. This walk arrives at a final vertex $v \in V_n$ that is either "accept" ($v \in A$) or "reject" ($v \notin A$).*

It is useful to view each layer transition as a matrix. Identify each layer with $[w]$. For $t \in [n]$ and $b \in \{0,1\}$, let $P_{t,b} \in \{0,1\}^{w \times w}$ be the matrix whose (i,j)-entry is 1 iff the edge labeled b from the i-th vertex of V_{t-1} goes to the j-th vertex of V_t.

Definition 2.2 (Permutation BP)**.** *A read-once branching program P is called a permutation BP if, for all $t \in [n]$ and $b \in \{0,1\}$, the matrix $P_{t,b}$ is a permutation matrix.*

From now on, fix some width-w length-n BP P. We define

$$T_t = \frac{P_{t,0} + P_{t,1}}{2},$$

which is the random-walk operator associated with layer t. We will frequently consider sub-programs corresponding to intervals of the original BP. For an interval $I = \{a, \ldots, b-1\} \subseteq [n]$ and an input string $x \in \{0,1\}^I$, define $P_I(x) = P_{a,x_a} \cdots P_{b-1,x_{b-1}}$. The corresponding uniform random-walk operator is

$$T_I = \mathbb{E}_{x \in \{0,1\}^I}[P_I(x)] = T_a T_{a+1} \cdots T_{b-1}. \tag{2.1}$$

Let $A \subseteq V_n$ denote the set of accepting states in the n-th layer, and let $s \in V_0$ denote the starting state. The value of the program on input $x \in \{0,1\}^n$ is then given by $e_s^\top P_{[n]}(x) \mathbb{1}_A$, where $e_s \in \mathbb{R}^w$ is the standard basis vector corresponding to s, and $\mathbb{1}_A \in \mathbb{R}^w$ is the indicator vector of A. For brevity, we may also denote this value by $P(x)$. Similarly, the acceptance probability according to a truly uniform walk is given by $e_s^\top T_{[n]} \mathbb{1}_A$.

A PRG for permutation BPs maps short uniformly random seeds to inputs while approximately preserving the acceptance probability of every program.

Definition 2.3. *A function $G : \{0,1\}^s \to \{0,1\}^n$ is an ε-PRG for width-w length-n permutation branching programs if for every such program P, we have*

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

In this case we say that G ε-fools length-n width-w permutation BPs.

2.2 Expander Graphs

As mentioned, the PRG we use is the INW generator, which is constructed from a family of expander graphs. We begin by recalling the definition of an expander.

Definition 2.4 (Expander graph)**.** *Let $G = (V,E)$ be an undirected, d-regular graph, and let A be its adjacency matrix. We say that the graph G is a λ-expander if the spectral norm of the matrix $\frac{1}{d}A - J$ is at most λ, where J is the normalized all-ones matrix.*

For the INW generator, we need the following construction of an explicit family of expander graphs.

Theorem 2.5 (e.g., [MRSV19], Theorem 3.3)**.** *For every $n \in \mathbb{N}$ and $\lambda > 0$, there exists an explicit graph H_n on n vertices, which is a D-regular λ-expander with $D = (1/\lambda)^{\Theta(1)}$. Moreover, for any vertex x and a neighbor label $y \in [D]$, the y-th neighbor of x, which is denoted by $H_n(x,y)$, can be computed in space $O(\log(nD))$.*

We will use the following vector-valued version of the expander mixing lemma.

Lemma 2.6. *Let $H = (V,E)$ be an expander graph with second eigenvalue $\lambda \in (0,1)$. Let $\alpha, \beta : V \to \mathbb{R}^k$ be two functions such that $\mathbb{E}_{x\in V}[\alpha(x)] = \mathbb{E}_{y\in V}[\beta(y)] = 0$. Then,*

$$\left|\mathbb{E}_{(x,y)\in E}[\langle \alpha(x), \beta(y)\rangle]\right| \leq \lambda \left(\mathbb{E}_{x\in V}[\|\alpha(x)\|_2^2]\right)^{1/2} \left(\mathbb{E}_{y\in V}[\|\beta(y)\|_2^2]\right)^{1/2}.$$

The proof follows from the standard expander-mixing lemma; see Section A for the details.

2.3 The INW Generator

We now recall the INW construction. Fix parameters n, w, ε. We assume without loss of generality that $n = 2^h$, since otherwise the program can be padded with trivial layers. Let d be an integer parameter to be chosen later. For every $i = 1, 2, \ldots, h$, let H_i be a 2^d-regular λ-expander on the vertex set $\{0,1\}^{(i-1)d+1}$, given by Theorem 2.5.

The INW generator is defined recursively as follows. For $i \geq 0$, we define a function

$$G_i \colon \{0,1\}^{id+1} \to \{0,1\}^{2^i}.$$

For $G_0\colon \{0,1\} \to \{0,1\}$, we set $G_0(x) = x$. For $i \geq 1$, write an input to G_i as (x,y), where

$$x \in \{0,1\}^{(i-1)d+1} \qquad \text{and} \qquad y \in \{0,1\}^d.$$

Then,

$$G_i(x,y) = \big(G_{i-1}(x), G_{i-1}(H_i(x,y))\big).$$

The generator for length-n programs is therefore G_h, where recall that $h = \log_2 n$. Its seed length is $O(d \log n)$. We instantiate the construction with the expanders given by Theorem 2.5. Consequently, the overall seed length, as well as the space complexity of the generator, is $O(\log(1/\lambda) \cdot \log n)$.

We define the analogue of T_I (see Equation (2.1)) for an interval $I \subseteq [n]$, with respect to the pseudorandom walk induced by the INW generator. More precisely, for $i \geq 0$ and for an interval $I \subseteq [n]$ of size 2^i, define

$$\widetilde{T}_I = \mathbb{E}_x \left[P_I(G_i(x)) \right],$$

where the expectation is over a uniformly random seed x for G_i.

3 Proof of Main Result

In this section we prove the following theorem, which is the formal and complete statement of Theorem 1.1.

Theorem 3.1. *For every $n, w \in \mathbb{N}$ and for all $0 < \varepsilon \leq 1$ the expander-based INW-generator set with $\lambda = \frac{\varepsilon}{32w^3}$ ε-fools length-n width-w permutation BPs. It has seed length*

$$O(\log(1/\lambda) \cdot \log n) = O((\log(1/\varepsilon) + \log w) \cdot \log n).$$

The proof is organized as follows. In Section 3.1, we introduce the forward and backward weights and establish two useful properties that will be used later. In Section 3.2, we bound the error of the expander product in terms of these weights; this is the key estimate underlying our error analysis. Finally, in Section 3.3, we analyze the error across the recursion and show that it does not deteriorate from one level to the next, thereby proving Theorem 3.1.

3.1 Forward and Backward Weights

The main ingredient in our error analysis is the notion of the program's weight, following [BRRY14]. Our key insight is that, for permutation BPs, the weight of the *reversed* program can also be used in the bound. In this section, we define the notions of forward weight and backward weight, where the latter corresponds to the BRRY notion of weight, expressed in the language of operators. Throughout this section, all definitions are with respect to a fixed permutation branching program P.

Forward weights. Let $1 \le a \le b \le n+1$ be integers, let $I = \{a, a+1, \ldots, b-1\}$, and let $p = p_a \in \mathbb{R}^w$. The *forward propagation of p_a along the interval I* is the sequence of vectors $p_a, p_{a+1}, \ldots, p_b$ defined recursively, for $t = a, a+1, \ldots, b-1$, by

$$p_{t+1}^\top = p_t^\top T_t = p_t^\top \cdot \frac{P_{t,0} + P_{t,1}}{2}.$$

When p_a is a probability distribution, this is precisely the random walk induced by the program, restricted to the interval I.

The *forward weight at layer $t \in I$* is defined by

$$F_t(p) = \left\| p_t^\top P_{t,0} - p_{t+1}^\top \right\|_1 + \left\| p_t^\top P_{t,1} - p_{t+1}^\top \right\|_1 = \left\| p_t^\top (P_{t,0} - P_{t,1}) \right\|_1.$$

The *forward weight over the interval I* is

$$F_I(p) = \sum_{t=a}^{b-1} F_t(p).$$

Observe that if we write the interval I as the concatenation of two consecutive intervals $I = L \circ R$, then

$$F_I(p) = F_L(p) + F_R(T_L^\top p). \tag{3.1}$$

Backward weights. Let $1 \le a \le b \le n+1$ be integers, let $I = \{a, \ldots, b-1\}$, and let $q = q_b \in \mathbb{R}^w$. The *backward propagation of q_b along the interval I* is the sequence of vectors $q_b, q_{b-1}, \ldots, q_a$ defined recursively, for $t = b-1, b-2, \ldots, a$, by

$$q_t = T_t q_{t+1} = \frac{P_{t,0} + P_{t,1}}{2} \cdot q_{t+1}.$$

If q_b is a probability distribution, then this is the random walk corresponding to the reversed program, since

$$q_t^\top = q_{t+1}^\top T_t^\top = q_{t+1}^\top \cdot \frac{P_{t,0}^\top + P_{t,1}^\top}{2}.$$

The *backward weight at layer $t \in I$* is defined by

$$B_t(q) = \left\| P_{t,0} q_{t+1} - q_t \right\|_1 + \left\| P_{t,1} q_{t+1} - q_t \right\|_1 = \left\| (P_{t,0} - P_{t,1}) q_{t+1} \right\|_1.$$

The *backward weight over the interval I* is

$$B_I(q) = \sum_{t=a}^{b-1} B_t(q).$$

Similarly to Equation (3.1), if we write the interval I as the concatenation of two consecutive intervals $I = L \circ R$, then

$$B_I(q) = B_L(T_R q) + B_R(q). \tag{3.2}$$

Lemma 3.2. *Let P be a permutation branching program of width w. For every interval $I = \{a, \dots, b-1\}$ and every $p, q \in \mathbb{R}^w$, we have*

$$B_I(q) \le 2 \sum_{1 \le i < j \le w} |q_i - q_j| \le 2w \|q\|_1 ,$$
$$F_I(p) \le 2 \sum_{1 \le i < j \le w} |p_i - p_j| \le 2w \|p\|_1 .$$

The proof follows from Lemma 5 of [BRRY14], applied both to P and to the reversed program. We also use the observation that the proof of Lemma 5 applies verbatim to arbitrary vectors in $\mathbb{R}^w$; it is not essential that the vector entries are in $[0,1]$. See Section A.

Lemma 3.3. *Fix some permutation branching program P of length n and width w, and let $1 \le a \le b \le n+1$ be integers. Let $I = \{a, \dots, b-1\}$, and let $p, q \in \mathbb{R}^w$. Then, for every $x \in \{0,1\}^I$, we have*

$$\left\| P_I(x)^\top p - T_I^\top p \right\|_1 \le \frac{1}{2} F_I(p),$$
$$\|P_I(x) q - T_I q\|_1 \le \frac{1}{2} B_I(q).$$

Proof. We first prove the forward inequality. Let $p_a, p_{a+1}, \dots, p_b$ be the forward propagation of $p_a = p$ along I, so that $p_{t+1} = T_t^\top p_t$. For $t \in \{a, a+1, \dots, b\}$, define

$$r_t = \left(P_{a,x_a} P_{a+1,x_{a+1}} \cdots P_{t-1,x_{t-1}} \right)^\top p,$$

with the convention that $r_a = p$. Thus $r_b = P_I(x)^\top p$ and $p_b = T_I^\top p$. Moreover, $r_{t+1} = P_{t,x_t}^\top r_t$. Therefore, for every $t \in I$,

$$\begin{aligned} \|r_{t+1} - p_{t+1}\|_1 &= \left\| P_{t,x_t}^\top r_t - p_{t+1} \right\|_1 \\ &\le \left\| P_{t,x_t}^\top (r_t - p_t) \right\|_1 + \left\| P_{t,x_t}^\top p_t - p_{t+1} \right\|_1 \\ &= \|r_t - p_t\|_1 + \left\| P_{t,x_t}^\top p_t - p_{t+1} \right\|_1 \\ &= \|r_t - p_t\|_1 + \frac{1}{2} F_t(p_t), \end{aligned}$$

where we used that permutation matrices preserve the ℓ_1 norm. Since $r_a = p_a = p$, induction gives

$$\left\| P_I(x)^\top p - T_I^\top p \right\|_1 = \|r_b - p_b\|_1 \le \frac{1}{2} \sum_{t=a}^{b-1} F_t(p_t) = \frac{1}{2} F_I(p).$$

The claim for the backward inequality is proved in a similar way. □

3.2 Analyzing the Expander Product

In this subsection, we prove the following proposition, which is the key estimate in our error analysis.

Proposition 3.4. *Fix some permutation branching program of length n and width w. Let $p, q \in \mathbb{R}^w$, and let $I = L \circ R \subseteq [n]$ be an interval of size 2^i, where L and R are its left and right halves. Then*

$$\left| p^\top \left(\widetilde{T}_I - \widetilde{T}_L \widetilde{T}_R \right) q \right| \leq \lambda \cdot F_L(p) \cdot B_R(q).$$

Proof. Let $\Omega = \{0,1\}^{(i-1)d+1}$ be the seed space of G_{i-1}. For $x, z \in \Omega$, define

$$A_x = P_L(G_{i-1}(x)), \qquad B_z = P_R(G_{i-1}(z)).$$

Then

$$\widetilde{T}_L = \mathbb{E}_x A_x, \qquad \widetilde{T}_R = \mathbb{E}_z B_z, \qquad \widetilde{T}_L \widetilde{T}_R = \mathbb{E}_{x,z} A_x B_z,$$

where x, z are independent and uniform over Ω. Moreover,

$$\widetilde{T}_I = \mathbb{E}_{(x,z) \sim H_i} A_x B_z,$$

where $(x,z) \sim H_i$ means that x is uniform over Ω and z is a uniformly random neighbor of x in H_i. Define

$$\alpha_x = A_x^\top p - \widetilde{T}_L^\top p, \qquad \beta_z = B_z q - \widetilde{T}_R q.$$

Then, $\mathbb{E}_x \alpha_x = 0$ and $\mathbb{E}_z \beta_z = 0$. Moreover,

$$\begin{aligned} \langle \alpha_x, \beta_z \rangle &= p^\top (A_x - \widetilde{T}_L)(B_z - \widetilde{T}_R) q \\ &= p^\top \left(A_x B_z - \widetilde{T}_L B_z - A_x \widetilde{T}_R + \widetilde{T}_L \widetilde{T}_R \right) q. \end{aligned}$$

Taking expectation over $(x,z) \sim H_i$, and using that both marginals of this distribution are uniform over Ω, we get

$$\mathbb{E}_{(x,z) \sim H_i} \langle \alpha_x, \beta_z \rangle = p^\top (\widetilde{T}_I - \widetilde{T}_L \widetilde{T}_R) q. \tag{3.3}$$

Hence, by Lemma 2.6,

$$\left| p^\top (\widetilde{T}_I - \widetilde{T}_L \widetilde{T}_R) q \right| \leq \lambda \left(\mathbb{E}_x \|\alpha_x\|_2^2 \right)^{1/2} \left(\mathbb{E}_z \|\beta_z\|_2^2 \right)^{1/2}.$$

It remains to bound the two terms on the right-hand side. For every $x \in \Omega$,

$$\alpha_x = A_x^\top p - T_L^\top p + T_L^\top p - \widetilde{T}_L^\top p.$$

By Lemma 3.3,

$$\left\| A_x^\top p - T_L^\top p \right\|_1 \leq \frac{1}{2} F_L(p).$$

Also,

$$T_L^\top p - \widetilde{T}_L^\top p = \mathbb{E}_{x'}\left[T_L^\top p - A_{x'}^\top p\right],$$

and therefore, again by Lemma 3.3 and convexity,

$$\left\| T_L^\top p - \widetilde{T}_L^\top p \right\|_1 \le \frac{1}{2} F_L(p).$$

Thus

$$\|\alpha_x\|_2 \le \|\alpha_x\|_1 \le F_L(p).$$

Similarly, for every $z \in \Omega$,

$$\|\beta_z\|_2 \le \|\beta_z\|_1 \le B_R(q).$$

Substituting these bounds gives

$$\left| p^\top (\widetilde{T}_I - \widetilde{T}_L \widetilde{T}_R) q \right| \le \lambda F_L(p) B_R(q),$$

as required. □

3.3 Error Analysis

In this section we analyze the error propagation throughout the recursion.

Proposition 3.5. *Let $\lambda > 0$ be a bound on the second eigenvalues of the expander graphs in the expander-based INW construction. Fix a permutation branching program P of length n and width w. Let $c \ge 2$, and assume that*

$$\lambda \le \frac{1}{8c^2 w^3}.$$

Let $p, q \in \mathbb{R}^w$, and let $I \subseteq [n]$ be an interval of size 2^i. Define

$$\mathscr{E}_I(p,q) = \left| p^\top (\widetilde{T}_I - T_I) q \right|.$$

Then,

$$\mathscr{E}_I(p,q) \le c\lambda F_I(p) B_I(q).$$

Proof. We prove the proposition by induction on i. When $i = 0$, we have $\widetilde{T}_I = T_I$, and the claim is immediate. Assume therefore that $i \ge 1$, and that the claim holds for intervals of size 2^{i-1}.

Write $I = L \circ R$, where L and R are the left and right halves of I. Set

$$A = F_L(p), \qquad B = F_R(T_L^\top p), \qquad C = B_L(T_R q), \qquad D = B_R(q).$$

By the decomposition properties of forward and backward weights (see Equations (3.1) and (3.2))

$$F_I(p) = A + B, \qquad B_I(q) = C + D.$$

Thus it suffices to prove

$$\mathscr{E}_I(p,q) \le c\lambda(A+B)(C+D).$$

We decompose the error as

$$\begin{aligned} p^\top(\widetilde{T}_I - T_I)q &= p^\top(\widetilde{T}_I - \widetilde{T}_L\widetilde{T}_R)q && (3.4)\\ &\quad + p^\top\widetilde{T}_L(\widetilde{T}_R - T_R)q && (3.5)\\ &\quad + p^\top(\widetilde{T}_L - T_L)T_Rq. && (3.6) \end{aligned}$$

We bound the three terms separately.

For (3.4), Proposition 3.4 gives

$$\left|p^\top(\widetilde{T}_I - \widetilde{T}_L\widetilde{T}_R)q\right| \le \lambda AD.$$

For (3.6), the induction hypothesis applied to the interval L, with vectors p and T_Rq, gives

$$\left|p^\top(\widetilde{T}_L - T_L)T_Rq\right| \le c\lambda AC.$$

For (3.5), the induction hypothesis applied to the interval R, with vectors $\widetilde{T}_L^\top p$ and q, gives

$$\left|p^\top\widetilde{T}_L(\widetilde{T}_R - T_R)q\right| = \left|(\widetilde{T}_L^\top p)^\top(\widetilde{T}_R - T_R)q\right| \le c\lambda D\cdot F_R(\widetilde{T}_L^\top p). \tag{3.7}$$

We next compare $F_R(\widetilde{T}_L^\top p)$ with $F_R(T_L^\top p) = B$.

Let

$$\delta = \widetilde{T}_L^\top p - T_L^\top p.$$

Since both T_L and $\widetilde{T}_L$ are doubly stochastic, we have $\sum_{j=1}^w \delta_j = 0$. Let $r \in \{0,1\}^w$ be the indicator vector of the set of coordinates on which δ is nonnegative, namely

$$r_j = \begin{cases} 1, & \delta_j \ge 0,\\ 0, & \text{otherwise.}\end{cases}$$

Then

$$\|\delta\|_1 = 2\langle\delta, r\rangle.$$

Moreover,

$$\langle\delta, r\rangle = p^\top(\widetilde{T}_L - T_L)r.$$

Applying the induction hypothesis to the interval L, with vectors p and r, we obtain

$$|\langle\delta, r\rangle| \le c\lambda F_L(p)B_L(r).$$

By Lemma 3.2, and since $\|r\|_1 \le w$,

$$B_L(r) \le 2w\|r\|_1 \le 2w^2.$$

Hence

$$\|\delta\|_1 \le 4w^2 c\lambda A.$$

Using the subadditivity of $F_R(\cdot)$ and Lemma 3.2, we get

$$F_R(\widetilde{T}_L^\top p) \le F_R(T_L^\top p) + F_R(\delta) \le B + 2w\|\delta\|_1 \le B + 8w^3 c\lambda A.$$

Since $\lambda \le 1/(8c^2w^3)$, it follows that

$$F_R(\widetilde{T}_L^\top p) \le B + \frac{A}{c}.$$

Therefore,

$$\left| p^\top \widetilde{T}_L(\widetilde{T}_R - T_R)q \right| \le c\lambda DB + \lambda DA.$$

Combining the three bounds, we conclude that

$$\mathscr{E}_I(p,q) \le 2\lambda AD + c\lambda AC + c\lambda DB. \tag{3.8}$$

Since $c \ge 2$, this is at most

$$c\lambda(A+B)(C+D), \tag{3.9}$$

completing the induction. □

We are finally ready to prove our main result.

Proof of Theorem 3.1. Without loss of generality we may assume that n is a power of 2, since otherwise we may add at most n identity layers to the program. By applying Proposition 3.5 and Lemma 3.2 with $c = 2$, $q = \mathbb{1}_A$, $p = e_s$, we conclude

$$\left| p^\top \left(T_{[n]} - \widetilde{T}_{[n]} \right) q \right| \le c\lambda F_{[n]}(p) B_{[n]}(q) \le 4c\lambda w^2 |A| < \varepsilon.$$

□

4 The Forward–Backward Seminorm

In this short section we present a different perspective on the proof of Proposition 3.4. Namely, we view the proof as controlling the propagation of error through the recursion in terms of a family of seminorms[2] adapted to the underlying subprograms.

[2] A seminorm on a vector space is a function that satisfies the triangle inequality and homogeneity, just like a norm, but is allowed to vanish on nonzero vectors. Homogeneity means that scaling the input scales the value by the absolute value of the scalar: if ρ is the seminorm, then $\rho(\alpha x) = |\alpha|\rho(x)$ for every scalar α. In our setting, this means that the seminorm may assign value zero to a nonzero error matrix, namely one whose action is not detected by the relevant forward and backward test vectors.

Definition 4.1. *Fix a permutation branching program of length n and width w, and let $I \subseteq [n]$ be an interval. For a $w \times w$ real matrix Δ, define the forward–backward seminorm by*[3]

$$\|\Delta\|_{\mathrm{fb},I} = \sup_{p,q:\, F_I(p)B_I(q)>0} \frac{|p^\top \Delta q|}{F_I(p)B_I(q)}.$$

Thus, every interval I of the program induces its own seminorm, tailored to the behavior of the branching program on that interval. In other words, the error is measured with respect to the "geometry" induced by the corresponding subprogram.

With this terminology, Proposition 3.5 can be restated in the following form.

Proposition 4.2. *Fix a permutation branching program of length n and width w. Let $c \geq 2$, and assume that $\lambda \leq \frac{1}{8c^2w^3}$. Then, for every dyadic interval $I \subseteq [n]$,*

$$\|T_I - \widetilde{T}_I\|_{\mathrm{fb},I} \leq c\lambda.$$

This formulation highlights the conceptual content of the proof of Proposition 3.5. Rather than controlling error propagation throughout the recursion using a single fixed norm, such as the spectral norm or the induced ℓ_1 norm, the proof uses a different seminorm for each dyadic subprogram. These seminorms are chosen to match the forward and backward structure of the corresponding interval. The main point is that, with respect to these adapted error measures, the error does not deteriorate through the recursion beyond a universal bound of $O(\lambda)$ provided λ is taken sufficiently small.

4.1 Relation to the Seminorm of [CHL+23]

In this section, we discuss how our seminorm relates to the one used by Chen, Hoza, Lyu, Tal, and Wu [CHL+23] that builds on the singular-value approximation [APP+23]. Although the two seminorms are different and arise in different contexts—theirs is used for the non-black-box derandomization of regular ROBPs, whereas ours is used to control error throughout the INW recursion—we believe that the comparison is instructive.

Let $I = L \circ R$ correspond to one step of the INW recursion, and let Ω be the seed space of the child generator. Recall the notation introduced in the proof of Proposition 3.4. For $x, z \in \Omega$, let $A_x = P_L(G_{i-1}(x))$, $B_z = P_R(G_{i-1}(z))$, and recall that $\widetilde{T}_L = \mathbb{E}_x[A_x]$, $\widetilde{T}_R = \mathbb{E}_z[B_z]$. For $p, q \in \mathbb{R}^w$, we defined $\alpha_x = A_x^\top p - \widetilde{T}_L^\top p$, $\beta_z = B_z q - \widetilde{T}_R q$. In Equation (3.3), we showed that

$$\mathbb{E}_{(x,z)\sim H_i}\langle \alpha_x, \beta_z \rangle = p^\top (\widetilde{T}_I - \widetilde{T}_L \widetilde{T}_R) q.$$

[3] If the supremum in the definition is infinite or taken over an empty set, we define the seminorm to be ∞.

Invoking Lemma 2.6, we then deduced that

$$\left|p^\top(\widetilde{T}_I - \widetilde{T}_L\widetilde{T}_R)q\right| \le \lambda\left(\mathbb{E}_x\|\alpha_x\|_2^2\right)^{1/2}\left(\mathbb{E}_z\|\beta_z\|_2^2\right)^{1/2}. \tag{4.1}$$

At this point, our discussion departs from the proof of Proposition 3.4.

Since A_x and B_z are permutation matrices, a direct calculation shows that the two variances appearing on the right-hand side of Equation (4.1) are exactly

$$\mathbb{E}_x\|\alpha_x\|_2^2 = p^\top(I - \widetilde{T}_L\widetilde{T}_L^\top)p,$$
$$\mathbb{E}_z\|\beta_z\|_2^2 = q^\top(I - \widetilde{T}_R^\top\widetilde{T}_R)q.$$

For a matrix M, define its left and right "defect"[4] quantities by

$$D_M^-(p) := \left(p^\top(I - MM^\top)p\right)^{1/2}, \qquad D_M^+(q) := \left(q^\top(I - M^\top M)q\right)^{1/2}.$$

We can therefore rewrite Equation (4.1) as

$$\left|p^\top(\widetilde{T}_I - \widetilde{T}_L\widetilde{T}_R)q\right| \le \lambda D_{\widetilde{T}_L}^-(p)D_{\widetilde{T}_R}^+(q). \tag{4.2}$$

The forward and backward weights used in the proof of Proposition 3.4 yield upper bounds on these defect quantities. Although these bounds are coarser, they are more convenient for our purposes.

The quantities D^- and D^+ also underlie the program-dependent seminorm introduced in [CHL+23]. In the notation of their paper, for a walk matrix $W_{r\leftarrow\ell}$ they define the interval seminorm

$$\|u\|_{\ell\rightsquigarrow r}^2 := u^\top\left(I - W_{r\leftarrow\ell}^\top W_{r\leftarrow\ell}\right)u = \|u\|_2^2 - \|W_{r\leftarrow\ell}u\|_2^2. \tag{4.3}$$

Under our transpose convention, $W_{r\leftarrow\ell} = T_{[\ell,r)}^\top$, and hence

$$\|u\|_{\ell\rightsquigarrow r}^2 = D_{T_{[\ell,r)}}^-(u)^2.$$

Their seminorm, which they call the F-seminorm—not to be confused with our notation F for the forward weight—aggregates these quantities over the dyadic intervals of the layered program:

$$\|x\|_{\mathsf{F}}^2 = \sum_{k=1}^{\log n}\sum_{j\in U_k}\left\|x^{[j]}\right\|_{j\rightsquigarrow j+2^k}^2, \tag{4.4}$$

where $U_k = \{0, 2^k, 2\cdot 2^k, \ldots, n\}$, with the convention adopted in their paper for the terminal layer. In Section 9 of their paper, the same defect quantities appear in the definition of singular-value approximation:

$$|y^\top(\widetilde{W} - W)x| \le \frac{\lambda}{4}\left(D_W^+(x)^2 + D_W^-(y)^2\right). \tag{4.5}$$

[4]We refer to these as the left and right defect quantities of M, since they measure the failure of $M^\top$ and M, respectively, to preserve Euclidean norm: $D_M^-(p)^2 = \|p\|_2^2 - \|M^\top p\|_2^2$, $D_M^+(q)^2 = \|q\|_2^2 - \|Mq\|_2^2$.

Thus, although the two seminorms are not identical, they are based on the same program-dependent notion of Euclidean dissipation. The seminorm of [CHL+23] is a one-sided, multiscale ℓ_2 aggregation over the entire layered space. By contrast, our seminorm is associated with a single interval and measures a two-sided bilinear error using the forward and backward ℓ_1 weights.

5 A Refined Analysis and a Convolution Form of the Error

In this section we present a more refined analysis of Proposition 3.5. The attentive reader may have noticed two sources of slack in the proof. The first is the mixture of ℓ_1 and ℓ_2 bounds. Indeed, the BRRY weight bound is stated in terms of the ℓ_1 norm, which in particular implies the corresponding ℓ_2 bound. By contrast, the proof of Proposition 3.5 uses spectral estimates throughout, except for the estimate on δ, which is given in terms of the ℓ_1 norm. The second source of slack is that, in the induction step from Equation (3.8) to Equation (3.9), the term BC is in fact redundant. It turns out that both sources of slack can be removed, leading to the more refined analysis presented in this section. While this improvement does not affect our main result, Theorem 1.1, we believe that this perspective is worth recording with an eye toward future work.

Before turning to the formal proof, we illustrate the idea behind the refined analysis. In Proposition 3.5, the error term

$$p^\top \widetilde{T}_L(\widetilde{T}_R - T_R)q$$

was bounded by applying the induction hypothesis to the interval R, with $\widetilde{T}_L^\top p$ as the left test vector. This required comparing $F_R(\widetilde{T}_L^\top p)$ with $F_R(T_L^\top p)$, the task we turned to after Equation (3.7). It is better to compare $T_L^\top p$ with its approximation $\widetilde{T}_L^\top p$ before passing them through the functional F_R. This can be done by introducing one additional hybrid, in addition to the three summands appearing in Equation (3.4), Equation (3.5), and Equation (3.6):

$$\widetilde{T}_L(\widetilde{T}_R - T_R) = T_L(\widetilde{T}_R - T_R) + (\widetilde{T}_L - T_L)(\widetilde{T}_R - T_R).$$

This introduces the second-order error term $(\widetilde{T}_L - T_L)(\widetilde{T}_R - T_R)$, but this term can be absorbed using Cauchy–Schwarz together with the induction hypothesis. With this in mind, we define the more accurate potential function as follows.

Definition 5.1 (Recursive forward–backward potential)**.** *Let I be a dyadic interval. We define a quantity $\Phi_I(p,q)$ recursively as follows. If $|I| = 1$, set*

$$\Phi_I(p,q) = 0.$$

If $I = L \circ R$, where L and R are the left and right halves of I, set

$$\Phi_I(p,q) = F_L(p)B_R(q) + \Phi_L(p, T_R q) + \Phi_R(T_L^\top p, q).$$

The potential $\Phi_I(p,q)$ is always upper bounded by the product $F_I(p)B_I(q)$, which is the quantity used in Proposition 3.5.

Claim 5.2. *For every dyadic interval I and every $p,q \in \mathbb{R}^w$,*

$$\Phi_I(p,q) \le F_I(p)B_I(q).$$

Proof. The claim is trivial when $|I| = 1$. Suppose $I = L \circ R$. Define the parameters A, B, C and D as in the proof of Proposition 3.5,

$$A = F_L(p), \qquad B = F_R(T_L^\top p), \qquad C = B_L(T_R q), \qquad D = B_R(q).$$

Then

$$F_I(p) = A + B, \qquad B_I(q) = C + D.$$

By the induction hypothesis,

$$\Phi_L(p, T_R q) \le AC, \qquad \Phi_R(T_L^\top p, q) \le BD.$$

Therefore,

$$\begin{aligned}
\Phi_I(p,q) &= AD + \Phi_L(p, T_R q) + \Phi_R(T_L^\top p, q) \\
&\le AD + AC + BD \\
&\le (A+B)(C+D) \\
&= F_I(p)B_I(q).
\end{aligned}$$

This completes the induction. □

We will also use the following notation. For a dyadic interval J, set

$$\kappa_F(J) = \sup_{\|u\|_2 = 1} F_J(u), \qquad \kappa_B(J) = \sup_{\|v\|_2 = 1} B_J(v).$$

Let

$$K = \sup_{J = L \circ R} \kappa_B(L)\kappa_F(R),$$

where the supremum is over all dyadic intervals J appearing in the recursion.

Proposition 5.3 (Refined forward–backward induction)**.** *Let $c \ge 2$, and assume that*

$$1 + c^2 \lambda K \le c.$$

Then, for every dyadic interval I and every $p,q \in \mathbb{R}^w$,

$$\left| p^\top (\widetilde{T}_I - T_I) q \right| \le c\lambda \Phi_I(p,q).$$

Proof. We prove the proposition by induction on $|I|$. If $|I| = 1$, then $\widetilde{T}_I = T_I$, so there is nothing to prove. Assume now that $I = L \circ R$, and that the claim holds for L and R. We decompose

$$\begin{aligned}\Delta_I &\triangleq \widetilde{T}_I - T_I \\ &= (\widetilde{T}_I - \widetilde{T}_L\widetilde{T}_R) + \Delta_L T_R + T_L\Delta_R + \Delta_L\Delta_R.\end{aligned}$$

We bound the four terms separately.

First, by Proposition 3.4,

$$\left|p^\top(\widetilde{T}_I - \widetilde{T}_L\widetilde{T}_R)q\right| \le \lambda F_L(p)B_R(q).$$

Second, by the induction hypothesis applied to the interval L,

$$\left|p^\top \Delta_L T_R q\right| \le c\lambda\Phi_L(p, T_R q).$$

Third, by the induction hypothesis applied to the interval R,

$$\left|p^\top T_L\Delta_R q\right| = \left|(T_L^\top p)^\top \Delta_R q\right| \le c\lambda\Phi_R(T_L^\top p, q).$$

It remains to bound the second-order term $p^\top\Delta_L\Delta_R q$. By Cauchy–Schwarz,

$$\left|p^\top\Delta_L\Delta_R q\right| \le \|\Delta_L^\top p\|_2 \cdot \|\Delta_R q\|_2.$$

We first bound $\|\Delta_L^\top p\|_2$. By duality,

$$\begin{aligned}\|\Delta_L^\top p\|_2 &= \sup_{\|r\|_2=1}\left|p^\top\Delta_L r\right| \\ &\le c\lambda \sup_{\|r\|_2=1} \Phi_L(p,r) \\ &\le c\lambda \sup_{\|r\|_2=1} F_L(p)B_L(r) \\ &\le c\lambda F_L(p)\kappa_B(L),\end{aligned}$$

where we used Claim 5.2 in the third line. Similarly,

$$\|\Delta_R q\|_2 \le c\lambda\,\kappa_F(R)B_R(q).$$

Therefore,

$$\left|p^\top\Delta_L\Delta_R q\right| \le c^2\lambda^2\kappa_B(L)\kappa_F(R)F_L(p)B_R(q).$$

By the definition of K, $\kappa_B(L)\kappa_F(R) \le K$. Hence the expander error and the second-order error together are at most

$$\left(\lambda + c^2\lambda^2 K\right)F_L(p)B_R(q) \le c\lambda F_L(p)B_R(q),$$

using the assumption $1 + c^2\lambda K \le c$. Combining the four bounds gives

$$\begin{aligned}\left|p^\top \Delta_I q\right| &\le c\lambda F_L(p)B_R(q) + c\lambda\Phi_L(p,T_R q) + c\lambda\Phi_R(T_L^\top p, q)\\ &= c\lambda\Phi_I(p,q).\end{aligned}$$

This completes the induction. □

Remark 5.4. By Lemma 3.2 and Cauchy–Schwarz, for every interval J and every $v \in \mathbb{R}^w$,

$$\begin{aligned}B_J(v) &\le 2\sum_{1\le i<j\le w}|v_i - v_j|\\ &\le 2\binom{w}{2}^{1/2}\left(\sum_{1\le i<j\le w}(v_i - v_j)^2\right)^{1/2}\\ &= 2\binom{w}{2}^{1/2}\left(w\|v\|_2^2 - \langle v, \mathbf{1}\rangle^2\right)^{1/2}\\ &\le \sqrt{2}\,w^{3/2}\|v\|_2.\end{aligned}$$

The same bound holds for $F_J(v)$. Thus

$$\kappa_F(J), \kappa_B(J) \le \sqrt{2}\,w^{3/2},$$

and consequently $K \le 2w^3$. In particular, the assumption $\lambda \le \frac{1}{8c^2w^3}$ from Proposition 3.5 is more than sufficient for $1 + c^2\lambda K \le c$, whenever $c \ge 2$.

The refined induction immediately implies Proposition 3.5, since Claim 5.2 gives $\Phi_I(p,q) \le F_I(p)B_I(q)$.

5.1 A Convolution Perspective

When expanding the recursion underlying the definition of Φ_I, we see it is exactly an ordered convolution of the local forward and backward weights, as discussed in this section.

Let $I = [a,b)$, let $p_a = p$, and let $q_b = q$. Define the propagated vectors

$$p_{t+1} = T_t^\top p_t \qquad \text{for } t = a, a+1, \dots, b-1,$$

and

$$q_t = T_t q_{t+1} \qquad \text{for } t = b-1, b-2, \dots, a.$$

For $t \in I$, define the local forward and backward weights

$$f_t = \left\|p_t^\top (P_{t,0} - P_{t,1})\right\|_1, \qquad b_t = \left\|(P_{t,0} - P_{t,1})q_{t+1}\right\|_1.$$

Thus

$$F_I(p) = \sum_{t=a}^{b-1} f_t, \qquad B_I(q) = \sum_{t=a}^{b-1} b_t.$$

Lemma 5.5 (Convolution identity)**.** *For every dyadic interval $I = [a,b)$ and every $p,q \in \mathbb{R}^w$,*

$$\Phi_I(p,q) = \sum_{a \le t < u < b} f_t b_u.$$

Proof. We prove the claim by induction on $|I|$. If $|I| = 1$, then both sides are zero.

Assume $I = L \circ R$, where $L = [a,m)$ and $R = [m,b)$. By definition,

$$\Phi_I(p,q) = F_L(p)B_R(q) + \Phi_L(p, T_R q) + \Phi_R(T_L^\top p, q).$$

The first term is

$$F_L(p)B_R(q) = \left(\sum_{t=a}^{m-1} f_t\right)\left(\sum_{u=m}^{b-1} b_u\right) = \sum_{\substack{a \le t < m \\ m \le u < b}} f_t b_u.$$

This accounts for all ordered pairs (t,u) separated by the split.

Next, since $T_R q = q_m$, the backward propagation inside L with terminal vector $T_R q$ is exactly the restriction of the above backward propagation to L. Therefore, by the induction hypothesis,

$$\Phi_L(p, T_R q) = \sum_{a \le t < u < m} f_t b_u.$$

Similarly, since $T_L^\top p = p_m$, the forward propagation inside R with initial vector $T_L^\top p$ is exactly the restriction of the above forward propagation to R. Hence

$$\Phi_R(T_L^\top p, q) = \sum_{m \le t < u < b} f_t b_u.$$

Adding the three sums gives

$$\Phi_I(p,q) = \sum_{a \le t < u < b} f_t b_u,$$

as claimed. □

References

[AKM+20] AmirMahdi Ahmadinejad, Jonathan Kelner, Jack Murtagh, John Peebles, Aaron Sidford, and Salil Vadhan. High-precision estimation of random walks in small space. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1295–1306. IEEE, 2020.

[APP+23] AmirMahdi Ahmadinejad, John Peebles, Edward Pyne, Aaron Sidford, and Salil Vadhan. Singular value approximation and sparsifying random walks on directed graphs. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 846–854. IEEE, 2023.

[BCG20] Mark Braverman, Gil Cohen, and Sumegha Garg. Pseudorandom pseudo-distributions with near-optimal error for read-once branching programs. *SIAM Journal on Computing*, 49(5):STOC18–242–STOC18–299, 2020.

[BDVY13] Andrej Bogdanov, Zeev Dvir, Elad Verbin, and Amir Yehudayoff. Pseudorandomness for width-2 branching programs. *Theory of Computing*, 9(7):283–293, 2013.

[BRRY14] Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. Pseudorandom generators for regular branching programs. *SIAM Journal on Computing*, 43(3):973–986, 2014.

[CHL+23] Lijie Chen, William M. Hoza, Xin Lyu, Avishay Tal, and Hongxun Wu. Weighted pseudorandom generators via inverse analysis of random walks and shortcutting. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1224–1239. IEEE, 2023.

[CW26] Kuan Cheng and Ruiyang Wu. Weighted pseudorandom generators for read-once branching programs via weighted pseudorandom reductions. In *Annual Symposium on Discrete Algorithms (SODA)*, pages 3423–3458. ACM-SIAM, 2026.

[De11] Anindya De. Pseudorandomness for permutation and regular branching programs. In *Computational Complexity Conference (CCC)*, pages 221–231. IEEE, 2011.

[HH24] Pooya Hatami and William Hoza. Theory of unconditional pseudorandom generators. *Foundations and Trends in Theoretical Computer Science*, 16(1-2):1–210, 2024.

[HL25] William M. Hoza and Zelin Lv. On sums of INW pseudorandom generators. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 67–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2025.

[HPV21] William M. Hoza, Edward Pyne, and Salil Vadhan. Pseudorandom generators for unbounded-width permutation branching programs. In *Innovations in Theoretical Computer Science Conference (ITCS)*, pages 7:1–7:20. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[HPV24] William M. Hoza, Edward Pyne, and Salil Vadhan. Limitations of the Impagliazzo–Nisan–Wigderson pseudorandom generator against permutation branching programs. *Algorithmica*, 86(10):3153–3185, 2024.

[INW94] Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *Annual Symposium on Theory of Computing (STOC)*, pages 356–364. ACM, 1994.

[KNP11] Michal Koucký, Prajakta Nimbhorkar, and Pavel Pudlák. Pseudorandom generators for group products. In *Annual Symposium on Theory of Computing (STOC)*, pages 263–272. ACM, 2011.

[LRTV09] Shachar Lovett, Omer Reingold, Luca Trevisan, and Salil Vadhan. Pseudorandom bit generators that fool modular sums. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 615–630. Springer, 2009.

[LV25] Chin Ho Lee and Emanuele Viola. Pseudorandom Bits for Non-Commutative Programs. In *Computational Complexity Conference (CCC)*, pages 9:1–9:22. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2025.

[MRSV19] Jack Murtagh, Omer Reingold, Aaron Sidford, and Salil Vadhan. Deterministic approximation of random walks in small space. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*, pages 42:1–42:22. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2019.

[MRT19] Raghu Meka, Omer Reingold, and Avishay Tal. Pseudorandom generators for width-3 branching programs. In *Annual Symposium on Theory of Computing (STOC)*, pages 626–637. ACM, 2019.

[Nis92] Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

[PV21] Edward Pyne and Salil Vadhan. Pseudodistributions that beat all pseudorandom generators. In *Computational Complexity Conference (CCC)*, pages 33:1–33:15. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[Ste12] Thomas Steinke. Pseudorandomness for permutation branching programs without the group theory. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 19, page 83, 2012.

[SZ95] Michael Saks and David Zuckerman. Unpublished manuscript, 1995.

A Missing Proofs

For completeness, in this section we include some missing proofs. We begin with Lemma 2.6, which, as mentioned, follows from the standard expander-mixing lemma.

Proof of Lemma 2.6. Write $\alpha = (\alpha_1, \ldots, \alpha_k), \beta = (\beta_1, \ldots, \beta_k) : V \to \mathbb{R}^k$. By the standard expander mixing lemma, for all i we have

$$\left|\mathbb{E}_{(x,y)\in E}[\alpha_i(x)\beta_i(y)]\right| \leq \lambda \left(\mathbb{E}_{x\in V}[\alpha_i(x)^2]\right)^{1/2} \left(\mathbb{E}_{y\in V}[\beta_i(y)^2]\right)^{1/2}.$$

Thus,

$$\begin{aligned}
\left|\mathbb{E}_{(x,y)\in E}[\langle \alpha(x), \beta(y)\rangle_{\mathbb{R}^k}]\right| &= \left|\mathbb{E}_{(x,y)\in E}\left[\sum_{i=1}^{k} \alpha_i(x)\beta_i(y)\right]\right| \\
&\leq \sum_{i=1}^{k} \left|\mathbb{E}_{(x,y)\in E}[\alpha_i(x)\beta_i(y)]\right| \\
&\leq \lambda \sum_{i=1}^{k} \left(\mathbb{E}_{x\in V}\left[\alpha_i(x)^2\right]\right)^{1/2} \left(\mathbb{E}_{y\in V}\left[\beta_i(y)^2\right]\right)^{1/2}.
\end{aligned}$$

By the Cauchy–Schwarz inequality, the above is bounded by

$$\lambda \left(\mathbb{E}_{x\in V} \|\alpha(x)\|_2^2\right)^{1/2} \left(\mathbb{E}_{y\in V} \|\beta(y)\|_2^2\right)^{1/2},$$

as needed. □

We include the proof of Lemma 3.2. As noted above, the proof follows directly from Lemma 5 of [BRRY14].

Proof of Lemma 3.2. We first prove the bound for $B_I(q)$. Consider the subprogram induced by the interval $I = [a,b)$, and label the vertices in layer b by the entries of $q = q_b$. Extend these labels backwards by setting

$$q_t = T_t q_{t+1} = \frac{P_{t,0} + P_{t,1}}{2} q_{t+1}$$

for $t = b-1, b-2, \ldots, a$. Thus the label of each vertex is the average of the labels of its two children. Since the branching program is a permutation branching program, this subprogram is regular.

The weight of this evaluation program, in the sense of [BRRY14], is exactly

$$\sum_{t=a}^{b-1} \left(\left\|P_{t,0}q_{t+1} - q_t\right\|_1 + \left\|P_{t,1}q_{t+1} - q_t\right\|_1\right) = B_I(q).$$

Lemma 5 of [BRRY14] therefore gives

$$B_I(q) \le 2 \sum_{1 \le i < j \le w} |q_i - q_j|.$$

Although Lemma 5 of [BRRY14] is stated for labels in $[0,1]$, this entails the displayed bound for arbitrary $q \in \mathbb{R}^w$ by an affine rescaling. Indeed, if q is not constant, set

$$\widehat{q} = \frac{q - \alpha \mathbb{1}}{\beta - \alpha}, \qquad \alpha = \min_i q_i, \quad \beta = \max_i q_i.$$

Since $T_t \mathbb{1} = \mathbb{1}$ for every t, the corresponding propagated labels satisfy

$$\widehat{q}_t = \frac{q_t - \alpha \mathbb{1}}{\beta - \alpha}.$$

Hence all edge weights, and also all pairwise differences in the terminal layer, are scaled by the same factor $\beta - \alpha$. Applying the $[0,1]$ version to $\widehat{q}$ and scaling back gives the desired bound. If q is constant, then both sides are zero.

The bound for $F_I(p)$ follows by applying the same argument to the reversed permutation branching program on the interval I. In the reversed program, the transition matrices are $P_{t,0}^\top$ and $P_{t,1}^\top$. Since the matrices $P_{t,0}$ and $P_{t,1}$ are permutation matrices, their transposes are their inverses and are again permutation matrices. The backward propagation in the reversed program is precisely

$$p_{t+1} = T_t^\top p_t = \frac{P_{t,0}^\top + P_{t,1}^\top}{2} p_t,$$

which is the forward propagation in the original program. Moreover, the corresponding edge weight at layer t is

$$\left\| P_{t,0}^\top p_t - p_{t+1} \right\|_1 + \left\| P_{t,1}^\top p_t - p_{t+1} \right\|_1 = F_t(p_t).$$

Therefore Lemma 5 of [BRRY14], applied to the reversed program, gives

$$F_I(p) \le 2 \sum_{1 \le i < j \le w} |p_i - p_j|.$$

Finally, for any $v \in \mathbb{R}^w$,

$$\sum_{1 \le i < j \le w} |v_i - v_j| \le \sum_{1 \le i < j \le w} (|v_i| + |v_j|) = (w-1) \|v\|_1 \le w \|v\|_1 .$$

Applying this with $v = q$ and $v = p$ gives the two remaining inequalities. □

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