Electronic Colloquium on Computational Complexity, Report No. 150 (2022)

Near-Optimal Derandomization of Medium-Width Branching Programs

Aaron (Louie) Putterman[*]
Harvard University
aputterman@g.harvard.edu

Edward Pyne[†]
MIT
epyne@mit.edu

November 7, 2022

Abstract

We give a deterministic white-box algorithm to estimate the expectation of a read-once branching program of length n and width w in space

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

In particular, we obtain a nearly optimal space $\widetilde{O}(\log n)$ derandomization of programs up to width $w = 2^{\sqrt{\log n}}$. Previously, the best known space complexity for this problem was

$$O\left(\min\{\log n \cdot \log w, \quad \log^{3/2} n + \sqrt{\log n} \cdot \log w\}\right)$$

via the classic algorithms of Savitch (JCSS 1970) and Saks and Zhou (JCSS 1999), which only achieve space $\widetilde{O}(\log n)$ for $w = \text{polylog}(n)$.

We prove this result by showing that a variant of the Saks-Zhou algorithm developed by Cohen, Doron, and Sberlo (ECCC 2022) still works without executing one of the steps in the algorithm, the so-called "random shift step." This allows us to extend their algorithm from computing the nth power of a $w \times w$ stochastic matrix to multiplying n *distinct* $w \times w$ stochastic matrices with no degradation in space consumption. In the regime where $w \geq n$, we also show that our approach can achieve parameters matching those of the original Saks-Zhou algorithm (with no loglog factors). Finally, we show that for $w \leq 2^{\sqrt{\log n}}$, an algorithm even simpler than our algorithm and that of Saks and Zhou achieves space $O(\log^{3/2} n)$.

Keywords: pseudorandomness, space-bounded computation

[*]Supported under the Simons Investigator Fellowship of Boaz Barak, NSF grant DMS-2134157, DARPA grant W911NF2010021, and DOE grant DE-SC0022199.

[†]Supported by an Akamai Presidential Fellowship.

ISSN 1433-8092

1 Introduction

There have been over four decades of work towards derandomizing space-bounded computation, i.e. proving $\mathbf{BPL} = \mathbf{L}$. A central, extensively studied problem that is "$\mathbf{BPL}$-complete" is to estimate the expectation of a read-once branching program of length n and width $w = n$. A long line of research has attacked $\mathbf{BPL} = \mathbf{L}$ via pseudorandom generators and other black-box derandomization techniques [Nis92, INW94, NZ96, Arm98, BCG20, HZ20].

Here, however, our focus is on the *white-box* setting, where the problem is equivalent to computing an approximate product of n stochastic matrices $M_1, \ldots, M_n \in \mathbb{R}^{w\times w}$, where M_i corresponds to the transition probabilities of the branching program from layer $i-1$ to layer i. From the classical work of Savitch [Sav70] it is easy to derive a space $O\left(\log^2 n\right)$ algorithm for this problem (where $w = n$) via a divide and conquer approach.

In 1995, Saks and Zhou gave a breakthrough algorithm that achieved space $O(\log^{3/2} n)$ for the same problem:

Theorem 1.1 ([SZ99])**.** *There is a deterministic algorithm such that given a stochastic $M_1, \ldots, M_n \in \mathbb{R}^{w\times w}$, the algorithm computes a $1/n$-approximation to $M_1 \cdots M_n$. The algorithm runs in space*

$$O\left(\log^{3/2} n + \sqrt{\log n} \cdot \log w\right).$$

In 2022, Cohen, Doron, and Sberlo gave an algorithm that improved on Saks and Zhou in the case where $w \ll n$, and we add the additional constraint that all the stochastic matrices are equal:

Theorem 1.2 ([CDS22])**.** *There is a deterministic algorithm such that given a stochastic matrix $M \in \mathbb{R}^{w\times w}$, the algorithm computes a $1/n$-approximation to M^n. The algorithm runs in space*

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

Unfortunately, Cohen et al.'s result does not seem to improve on Saks and Zhou for computing the product of n *distinct* $w \times w$ stochastic matrices, and thus does not improve on the space complexity of derandomizing branching programs. We remark that for branching programs of width $w = o(n)$, the restriction that all transition functions are the same is severe: there are simple functions computable by ordered branching programs of width 2 that cannot be computed by "single transition" ordered programs of width $n/2 - 1$ (see Appendix B). Thus, for estimating the expectation of a width w, length n read-once branching program, the best space complexity remains

$$O\left(\min\left\{\log n \cdot \log w, \quad \log^{3/2} n + \sqrt{\log n} \cdot \log w\right\}\right)$$

from [Sav70, SZ99].

1.1 Main Result

We achieve near-optimal space complexity for derandomizing branching programs up to $w = \exp(\log^{1/2} n)$, and improve on known results for all $w \in [\log^{\omega(1)} n, \exp(\log^{.99} n)]$.

Theorem 1.3. *There is a deterministic algorithm such that given $n, w \in \mathbb{N}$ and a read-once branching program B of length n and width w, the algorithm approximates the acceptance probability of B up to error $1/n^2$. The algorithm runs in space*

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

This result can be viewed as an attack on **BPL** vs **L** from a different direction. While Saks and Zhou (with a further improvement from Hoza) [SZ99, Hoz21] decrease the space required to derandomize a width n, length n branching program, we increase the maximum width w such that we can near-optimally derandomize a width w, length n branching program. This can be thought of as a white-box analogue of trying to "build up" better PRGs by starting with the constant width regime [CHHL19, MRT19, FK18].

We state our main result in terms of computing an approximate product of stochastic matrices. For this theorem, we without loss of generality assume $w \le n$.

Theorem 1.4. *There is an algorithm $\mathcal{A}_1$ that, given $n, w \in \mathbb{N}$ with $w \le n$ and arbitrary stochastic matrices $M_1, \ldots, M_n \in \mathbb{R}^{w \times w}$, returns a matrix $\widetilde{M}$ satisfying*

$$\left\| M_1 \cdots M_n - \widetilde{M} \right\| \le 1/n^2.$$

Furthermore, the space complexity of the algorithm is

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

We note that where we may assume additional structure on $M_1, \ldots, M_n$, such as them being the random walk matrices of undirected or Eulerian graphs, we have existing near-optimal algorithms via a line of work on the Laplacian paradigm [MRSV17, MRSV19, AKM+20].

1.2 Overview of Prior Work

As our work builds on the approach of Saks and Zhou and Cohen, Doron, and Sberlo, we give a high-level presentation of their approaches before describing our improvements.

The Work of Saks and Zhou [SZ99] We first present the approach of [SZ99]. For simplicity we assume we are approximating the nth power of a single stochastic matrix $M \in \mathbb{R}^{n \times n}$.

The insight of Saks and Zhou was to divide the problem of computing the nth power of a stochastic matrix M into $\sqrt{\log n}$ iterations of computing the $\ell := 2^{\sqrt{\log n}}$th power of matrices $M, M^\ell, M^{\ell^2}, \ldots$. In each iteration, they used the Nisan generator, which we denote $\texttt{NIS}$. $\texttt{NIS}$ has an "offline" seed h, which in this case can be of length $O(\log^{3/2} n)$, and an "online" seed of length $O(\log n)$, such that for every fixed offline seed h the generator $\texttt{NIS}_h$ can be evaluated in space $O(\log n)$. Moreover, with high probability over h the generator $\texttt{NIS}_h$ produces a good approximation (in expectation over the *online* seed) of the ℓth power of a fixed stochastic matrix. They then reused the offline seed throughout the $\sqrt{\log n}$ levels of recursion, at each step using $\texttt{NIS}_h$ to approximate the ℓth power of the previous level. Thus, the algorithm uses $O(\log n)$ bits of space for the online seed at each of the $\sqrt{\log n}$ levels, and a single offline seed of length $O(\log^{3/2} n)$.

However, there is an issue with this construction as stated. The approximation $\widetilde{M} \approx M^\ell$ output by $\texttt{NIS}_h$ can (of course) depend on h. But then the claim that $\texttt{NIS}_h$ is good at approximating powers of $\widetilde{M}$ with high probability does not necessarily hold, as we reuse the same offline seed h. To avoid this, between each level of recursion Saks and Zhou randomly shift and round the matrix $\widetilde{M}$. By doing so, they ensure that with high probability the rounded matrix equals the relevant "true" power, and so $\texttt{NIS}_h$ is good for it with high probability. This results in an additional $O(\log n)$ bits of randomness per level, which is tolerable in their regime.

The Work of Cohen, Doron, and Sberlo [CDS22] We now discuss the subsequent work of [CDS22], which achieves improved space where $w \ll n$ and all matrices are the same. For simplicity, assume for now that $w = 2^{\sqrt{\log n}}$. Their work contributed the crucial insight that one could use the Nisan generator to approximate $2^{\sqrt{\log n}}$th powers to accuracy $2^{-c\sqrt{\log n}} \gg 1/n$. This reduced the offline seed to $O(\log n)$, and the online seed to $O(\sqrt{\log n})$.

While this larger error would degrade too much if we let it accumulate over the $\sqrt{\log n}$ levels, they instead used Richardson Iteration to reduce the error back to $1/n$ between each level. Richardson Iteration is a method for improving the accuracy of an approximate power with near-optimal space complexity that has seen several recent applications in the space bounded regime [AKM+20, PV21, CDR+21, CDS22].

Unfortunately, their approach seems to require that we are approximately powering a single $w \times w$ matrix, and thus cannot be used to derandomize read-once branching programs of width $w \ll n$ and length n. The reason for this relates to the random shifts used in the construction. Cohen et al. prove that the random shifts can be sampled with $O\left(\sqrt{\log n} + \log w\right)$ bits per level. Intuitively, this is because for the ith level, there are w^2 distinct values in the true power M^{ℓ^i}, and we must ensure that with high probability our rounding threshold does not lie near one of these values. But if instead there are n distinct base matrices, there could be nw^2 "bad" values per level and so we must invest $O(\log n)$ bits per level in the shift, leading to an eventual space consumption of $O(\log^{3/2} n)$.[1]

The other approach to dealing with distinct transition matrices, the one taken in the original paper of Saks and Zhou and other works [SZ99, AKM+20, PV21, CDR+21], is that given the $w \times w$ transition matrices $M_1, \ldots, M_n$, first embed them as the off-diagonal elements in a nearly $nw \times nw$ matrix:

$$M = \begin{bmatrix} 0 & M_1 & 0 & \ldots & 0 \\ 0 & 0 & \ddots & & \vdots \\ 0 & \ldots & 0 & M_{n-1} & 0 \\ 0 & & & 0 & M_n \\ 0 & & \ldots & & 0 \end{bmatrix} \tag{2}$$

Then the $(1, n+1)$ block of M^n equals $M_1 \cdots M_n$, so an approximate nth power of M can be used to read off an approximation to the product. However, we are unable to take advantage of this method for computing the product of n distinct $w \times w$ matrices when $w = n^{o(1)}$. This is because this block matrix will necessarily be of size at least $n \times n$, and thus result in space consumption $O(\log^{3/2} n)$.

1.3 Our Approach

We obtain Theorem 1.4 by showing that the random shift step can be *eliminated* from the algorithm of [CDS22]. The reason for random rounding in [SZ99, CDS22] is the approximate product $\widetilde{M} \approx M^\ell$, even at iteration 1, can depend on the offline seed h to the Nisan PRG, and so we cannot say that $\texttt{NIS}_h$ is good for powering this new matrix with high probability. The solution of prior works is to randomly shift $\widetilde{M}$ before rounding, such that is is *exactly* equal to (the shifted and rounded version of) M^ℓ, and then apply $\texttt{NIS}_h$. Since M^ℓ is defined without reference to h, we can say that $\texttt{NIS}_h$ is good for it with high probability.

We take a different approach, by developing a more sophisticated analysis of the error incurred when using $\texttt{NIS}_h$ to approximate powers. Given a stochastic matrix $M \in \mathbb{R}^{w \times w}$, we feed M through

[1] They also gave an algorithm based on the Cayley Hamilton theorem that likewise does not seem to generalize to distinct matrices.

a canonicalizer $\mathtt{C_t}$ that outputs a branching program B such that walks on B are approximately distributed according to M. We can then estimate walk probabilities using a suitable pseudorandom generator (in this case, $\mathtt{NIS}_h$). More precisely, from state $v \in [w]$ the canonicalizer assigns the edge with label $\sigma \in [2^t]$ to the state k such that

$$\sum_{j=1}^{k-1} M_{v,j} \leq 2^{-t} \cdot \sigma < \sum_{j=1}^{k} M_{v,j}.$$

We observe that given M, M' that are close (say, within γ in ℓ_∞ distance), their canonicalizations $B := \mathtt{C_t}[M], B' := \mathtt{C_t}[M']$ are close *as labeled branching programs.* In particular, from every state v, most of the edges from state v have exactly the same destination in B and B'. Moreover, the only edges that differ must be those assigned in places where the partial sums of rows of M and M' differ. These locations can be (roughly) determined knowing only the partial sums of M and a bound on the ℓ_∞ distance of M' to M. Thus, there is is a set of roughly $2^t\gamma$ "boundary" edges from each state such that for *every* M' such that $\|M - M'\| \leq \gamma/w$, every difference between B and $B' := \mathtt{C_t}[M']$ will occur on these edges. Note that this set depends only on M and γ, and not on the specific M'. Thus, letting E be the program that accepts if we traverse a boundary edge, a generator $\mathtt{NIS}_h$ that fools B and E must also fool $B' = \mathtt{C_t}[M']$, even for M' that depend on the offline seed seed h. We give the formal statement of this result here, and prove it in Section 3.1:

Theorem 1.5. *Fix $t \in \mathbb{N}$ and $\gamma \geq 2^{-t}$ and let $M_1, \ldots, M_\ell \in \mathbb{R}^{w \times w}$ be sub-stochastic matrices. Suppose $\mathtt{GEN} : \{0,1\}^d \to [2^t]^\ell$ is ε-good for $\mathtt{C_t}[M_1, \ldots, M_\ell]$ and E, where E is defined in Lemma 3.5 only in terms of $M_1, \ldots, M_\ell$ and t and γ. Then for every sub-stochastic $\widetilde{M}_1, \ldots, \widetilde{M}_\ell \in \mathbb{R}^{w \times w}$ where $\left\|M_i - \widetilde{M}_i\right\| \leq \gamma$ for every i, $\mathtt{GEN}$ is $\rho := 6w\ell\gamma + 2\varepsilon$-good for $\mathtt{C_t}\left[\widetilde{M}_1, \ldots, \widetilde{M}_\ell\right]$.*

Since an ℓ_∞ error of magnitude γ can shift the value of all w partial sums by γ (and hence a $w \cdot \gamma$ fraction of edges could be allocated differently from each vertex), we obtain the promise that if M, M' are γ close, we fool $\mathtt{C_t}[M']$ up to error roughly $\gamma \cdot w$. While this is too much for the original algorithm of Saks and Zhou (except in the small-width regime, see Theorem 1.8), we can use the idea of Cohen, Doron, and Sberlo [CDS22] to reduce the error back to $O(\gamma)$ using Richardson iteration. In fact, we prove that their exact algorithm with the random shift step deleted computes a good approximation. This results in a particularly simple analysis of the final algorithm, which we view as an advantage of our approach.

Remark 1.6. We remark that our canonicalizer can be viewed as truncating the matrix to t bits of precision, then locally monotonizing the edges from each vertex (in that the state reached from vertex v is a non-decreasing function of the edge label σ, for every v). This local monotonization procedure has found several applications in the black box setting, both in analyzing PRGs and bounding the advantage of programs on the coin problem [BV10, CHRT18, MRT19, BGZ21]. While previous results used local monotonization in the analysis, we take advantage of the white-box setting to actually *construct* a locally monotone branching program and apply the PRG on it. In Theorem 1.8, we give a space $O(\log^{3/2} n)$ algorithm for derandomizing programs of width up to $\exp(\log^{1/2} n)$ for which this is the only non-black-box operation in the algorithm.

Decreasing the Failure Probability We must make one final modification to the algorithm of [CDS22] to handle distinct matrices. They use the Nisan PRG [Nis92] to approximate powers. Interestingly, our result does not seem to work with the Nisan PRG – we instead use the Nisan PRG combined with the "sampler trick" of Armoni [Arm98]. This is because we require the failure probability δ of the generator to be polynomially small in n (over the offline randomness), not merely w as in [CDS22], because there are nw distinct subprograms to approximate. For Nisan, this would force an *online* seed of length $O(\log n)$, which would require too much space.

To obtain an online seed of length $O\left(\sqrt{\log n} + \log w\right)$, we compose Nisan with an averaging sampler with doubly logarithmic dependence on δ in the online component. We note that evaluating the averaging sampler on an online seed (of length $O\left(\sqrt{\log n} + \log w\right)$) now takes workspace $O(\log n + \sqrt{\log n} \cdot \log w)$. However, since this workspace is purely used to produce the sampler output (and hence we do not read the input while using it) we can reuse a fixed $O(\log n + \sqrt{\log n} \cdot \log w)$ bits of workspace for these evaluations across all levels of recursion.

Interestingly, our argument relies on the sampler (and initial PRG) being an *averaging* sampler, because we use a sandwiching argument. The original argument of Saks and Zhou (and Cohen et al.) works more generally for a *weighted* PRG, and this was crucial for Hoza's improved derandomization of **BPL** [Hoz21]. We view whether our approach works for weighted PRGs to be an interesting open question.

We give a sketch of the our algorithm in the case that $M_1 = M_2 = \cdots = M_n$. For the formal description, see Algorithm 2.

Algorithm 1: Algorithm sketch

```
1 Draw a random seed $h$ and let $\texttt{NIS}_h$ be the generator with offline seed $h$.
2 Set $\widetilde{M}^0 = M_1$.
3 for $i = 1, \dots \sqrt{\log n}$ do
4     Canonicalize $\widetilde{M}^{i-1}$ to a degree $2^{c\sqrt{\log n}}$ branching program denoted $\texttt{C}_{\texttt{t}_1}\left[\widetilde{M}^{i-1}\right]$.
5     Use $\texttt{NIS}_h$ to approximate $\texttt{C}_{\texttt{t}_1}\left[\widetilde{M}^{i-1}\right]^{2^{\sqrt{\log n}}}$ up to error $2^{-c\sqrt{\log n}}$.
6     Use $\widetilde{O}(1)$ applications of the space efficient Richardson Iteration algorithm to reduce
        the error of the previous matrix to $1/n^c$ and call the matrix resulting from this step
        $\widetilde{M}^i$. This reduces the error sufficiently such that we can apply Nisan's generator on
        $\widetilde{M}^i$ despite the fact that this resulting matrix depends on $h$.
7 end
8 return $\widetilde{M}^{\sqrt{\log n}}$.
```

1.4 Other Results

Our approach allows us to eliminate the random shifts in the Saks-Zhou algorithm in a range of regimes, which we explicate here. First, by applying Richardson iteration to Theorem 1.4, we can boost our main result to arbitrarily low error:

Corollary 1.7. *Given $w, n \in \mathbb{N}$ with $w \leq n$ and $\varepsilon > 0$ and arbitrary stochastic matrices $M_1, \dots, M_n \in \mathbb{R}^{w \times w}$, there is an algorithm $\mathcal{A}_1'$ that returns a matrix $\widetilde{M}$ satisfying*

$$\left\| M_1 \cdots M_n - \widetilde{M} \right\| \leq \varepsilon.$$

Furthermore, the space complexity of the algorithm is

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

Furthermore, we show that for matrices that are not too wide, the naive approach of iteratively applying the Nisan generator (with $1/\operatorname{poly}(n)$ error) and canonicalizing works, even without Richardson Iteration. This does not improve on Savitch, but we include it as the resulting algorithm is particularly simple. In particular, the only non-black-box step is the canonicalization operation.

Theorem 1.8. *There is an algorithm $\mathcal{A}_2$ that, given $n \in \mathbb{N}$ and arbitrary stochastic matrices $M_1, \ldots, M_n \in \mathbb{R}^{w \times w}$ where $w \leq 2^{\sqrt{\log n}}$, returns a matrix $\widetilde{M}$ satisfying*

$$\left\| M_1 \cdots M_n - \widetilde{M} \right\| \leq 1/n^2.$$

Furthermore, the space complexity of the algorithm is $O\left(\log^{3/2} n\right)$.

Finally, we show that we can achieve space complexity matching Saks-Zhou (with no loglog factors) without random shifts, by interleaving approximate powers with a constant number of Richardson iterations.

Theorem 1.9. *There is an algorithm $\mathcal{A}_3$ that, given $n, w \in \mathbb{N}$ and a stochastic matrix $M \in \mathbb{R}^{w \times w}$, returns a matrix $\widetilde{M}$ satisfying*

$$\left\| M^n - \widetilde{M} \right\| \leq 1/w^2.$$

Furthermore, the space complexity of the algorithm is $O\left(\log^{3/2} n + \sqrt{\log n} \cdot \log w\right)$.

1.5 Organization

In Section 2 we define terms and recall the Nisan generator (with a modification to decrease its failure probability) and space-efficient Richardson iteration. In Section 3 we formally define the canonicalizer and prove Theorem 1.5. In Section 4 we apply Theorem 1.5 to prove our main result. In Section 5 we show that for small widths, an algorithm without shifts or Richardson iteration can achieve space $O(\log^{3/2} n)$. In Section 6 we show that we can achieve space matching that of Saks and Zhou for large widths, by replacing random shifts by a constant number of Richardson iterations per level.

2 Preliminaries

We first define several terms that we will use in the proofs.

- Let $M \in \mathbb{R}_{\geq 0}^{w \times w}$ be a **stochastic** (resp. **sub-stochastic**) matrix if every row sum is equal (resp. at most) 1. In the language of Markov chains, $M_{i,j}$ is the transition probability from state i to state j.
- Let $\| \cdot \|$ denote the ℓ_∞ matrix norm. We remark that none of our results are sensitive to the precise choice of norm.
- Let U_S denote the uniform distribution over elements of the set S. For $n \in \mathbb{N}$ let U_n be the uniform distribution over $\{0, 1\}^n$.

We recall the property that products of substochastic matrices approximate each other:

Claim 2.1. *Given sub-stochastic $A_1, \dots, A_\ell \in \mathbb{R}^{w \times w}$ and $B_1, \dots, B_\ell \in \mathbb{R}^{w \times w}$, if $\|A_i - B_i\| \le \delta$ for every i, then*

$$\|A_1 \cdots A_\ell - B_1 \cdots B_\ell\| \le \ell \cdot \delta.$$

We now formally define branching programs. We define them without reference to a distinguished start or accept state, as we will always require approximations of the walk probability from every state in layer 1 to every state in layer n.

Definition 2.2. A (read-once) **branching program** B of width w and length n with alphabet Σ consists of n functions $B_i : [w] \times [\Sigma] \to [w]$. We define the composition of branching programs $B := B_1 \cdots B_n$ in the natural way. For $x \in \Sigma^n$ we define

$$B[i,x] = (B_1 \cdots B_n)[i,x] := B_n[B_{n-1}[\dots B_2[B_1[i,x_1],x_2]\dots],x_{n-1}],x_n].$$

We view the expectation of a branching program as a stochastic matrix. Let $\overline{\mathbb{E}}[B]$ be the matrix where $\left(\overline{\mathbb{E}}[B]\right)_{i,j} = \Pr[B[i,U_{\Sigma^n}] = j]$. For a function $\texttt{GEN} : \{0,1\}^s \to \Sigma^n$, let $\overline{\texttt{GEN}}[B] = \overline{\mathbb{E}}[B \circ \texttt{GEN}]$.

We say a function $\texttt{GEN} : \{0,1\}^s \to \Sigma^n$ is ε-**good** for a branching program B if for every subprogram $B_{i..j}$ we have (truncating the output of GEN to its $j-i$ bit prefix):

$$\left\|\overline{\texttt{GEN}}[B_{i..j}] - \overline{\mathbb{E}}[B_{i..j}]\right\| \le \varepsilon.$$

2.1 Averaging Samplers and Error Reduction Primitives

We recall the family of pseudorandom generators to be used in our construction. We use the Nisan PRG composed with an averaging sampler. We use a formulation of Chattopadhyay and Liao [CL20].

Theorem 2.3 ([Nis92, RVW02, CL20]). *Given $n, w, |\Sigma| \in \mathbb{N}$ and $\varepsilon, \delta > 0$, there exists a generator $\texttt{NIS} : \{0,1\}^m \times \{0,1\}^d \to \Sigma^n$ such that for every length n, width w branching program B we have:*

$$\Pr_{h \leftarrow U_m}\left[\left\|\overline{\texttt{NIS}(h,\cdot)}[B] - \overline{\mathbb{E}}[B]\right\| \le \varepsilon\right] \ge 1 - \delta$$

and $m = O(\log(n)\log(nw|\Sigma|/\varepsilon) + \log(n/\delta))$ and $d = O(\log(nw|\Sigma|/\varepsilon) + \log\log(n/\delta))$. Equivalently, for every branching program B, with probability $1-\delta$ over h $\texttt{NIS}_h := \texttt{NIS}(h,\cdot)$ is ε-good for B. Furthermore, $\texttt{NIS}_h(x)$ can be evaluated in space $O(m)$ given two-way read-only access to the offline seed h.

For completeness, we provide a proof of this in Appendix A.

We can transform this generator into a space-efficient algorithm for approximating walks in branching programs, and this is the formulation we will use in the proof:

Lemma 2.4 ([Nis92, CL20, CDS22]). *Given $n, w, |\Sigma| \in \mathbb{N}$ and $\varepsilon, \delta > 0$, there exists an algorithm $\texttt{NIS}_{h,\varepsilon,\delta}$ that gets as (read only) input a branching program $B := B_1 \cdots B_n : \Sigma^n \to \{0,1\}^{w \times w}$ of width w, accuracy and confidence parameters $\varepsilon, \delta > 0$, and $h \in \{0,1\}^m$ where $m = O(\log(n)\log(nw|\Sigma|/\varepsilon) + \log(n/\delta))$. Furthermore:*

- *The algorithm runs in space $O(\log(nw|\Sigma|/\varepsilon) + \log\log(n/\delta))$, plus an additional $O(m + \log(n/\delta))$ space for producing the output of the sampler, during which it does not touch the input or output tapes.*

- *The algorithm outputs*
$$\left\{ \underset{i\to j}{\overline{\texttt{NIS}}} [B] \right\}_{i,j\in[n]}$$
where $\underset{i\to j}{\overline{\texttt{NIS}}} [B]$ *is a substochastic matrix. For every* $i < j$ *and* $u, v \in [w]$, $\left(\underset{i\to j}{\overline{\texttt{NIS}}} [B]\right)_{u,v}$ *is the fraction of outputs of* $\texttt{NIS}_{h,\varepsilon,\delta}(x)$ *that reach state* v *in layer* j *from state* u *in layer* i*. If* $B_i = B_j$ *for all* i, j *we assume without loss of generality the algorithm returns* $\left\{ \underset{1\to k}{\overline{\texttt{NIS}}} [B] \right\}_{k\in[n]}$.

- *For every branching program* B*, with probability* $1 - \delta$ *over* $h \leftarrow U_m$ *we have that for every* $i < j$,
$$\left\| \underset{i\to j}{\overline{\texttt{NIS}}} [B] - \overline{\mathbb{E}} [B_i \cdots B_j] \right\| \le \varepsilon.$$
We say that $\texttt{NIS}_{h,\varepsilon,\delta}$ *is* ε***-good*** *for* B *if this holds.*

Likewise, we recall the space-efficient Richardson Iteration algorithm used in prior work.

Lemma 2.5 ([AKM+20, CDR+21, PV21, CDS22])**.** *There exists an algorithm* $\texttt{R}$ *that, given sub-stochastic* $M_1, \dots, M_n \in \mathbb{R}^{w\times w}$ *and* $\left\{\widetilde{M_{i,j}}\right\}_{i<j}$ *and* $k \in \mathbb{N}$ *such that for all* $i < j$, $\left\|M_i \cdots M_j - \widetilde{M_{i,j}}\right\| \le 1/5n$ *returns a sub-stochastic matrix* $\texttt{R}(\{M_i\}_i, \{\widetilde{M_{i,j}}\}_{i,j}, k)$ *where each entry is represented by at most* t *bits of precision satisfying*
$$\left\|\texttt{R}\left(\{M_i\}_i, \{\widetilde{M_{i,j}}\}_{i,j}, k\right) - M_1 \cdots M_n\right\| \le 2nw^2 \cdot 2^{-k}.$$

Furthermore, $\texttt{R}$ *runs in space* $O(\log^2 k + \log(k)\log(nT))$ *where* T *is the maximum bit complexity of* $\{M_i\}, \{\widetilde{M_{i,j}}\}$ *In the case that* $M_i = M$ *for all* i*, we drop the* i *subscript without loss of generality.*

3 The Canonicalizer

We first formally define the canonicalizer. Informally, the canonicalizer is an algorithm that converts a substochastic matrix to a branching program, so that we can approximate its expectation using a pseudorandom generator. Our construction crucially relies on how the canonicalizer assigns edges, which we detail below:

Definition 3.1. Given $t \in \mathbb{N}$, there exists a ***t*-canonicalizer** $\texttt{C}_\texttt{t}$ that takes in a sub-stochastic matrix $M \in \mathbb{R}^{w\times w}$ with each entry represented by at most s bits and returns a branching program $\texttt{C}_\texttt{t}[M]$ on $w + 1$ states[2] with alphabet $\Sigma = [2^t]$. Furthermore, for $i, j, \sigma \in [w] \times [w] \times [2^t]$ the canonicalizer assigns $\texttt{C}_\texttt{t}[M][i, \sigma] = j$ if and only if
$$\sum_{k=1}^{j-1} M_{i,k} \le \sigma \cdot 2^{-t} < \sum_{k=1}^{j} M_{i,k}.$$

Furthermore, $\texttt{C}_\texttt{t}$ can be computed in space $O(\log(wts))$. We define $\texttt{C}_\texttt{t}[M_1, \dots, M_\ell] = \texttt{C}_\texttt{t}[M_1] \cdots \texttt{C}_\texttt{t}[M_\ell]$ for substochastic matrices $M_1, \dots, M_\ell$ in the natural way, where now the canonicalizer outputs a branching program of width $w + 1$ and length ℓ and the space complexity is $O(\log(wts\ell))$.

[2]Since in all cases this additional state will be immaterial, we implicitly pad M to be a (stochastic) $(w+1)\times(w+1)$ matrix and treat the transformation as width preserving.

We collect some basic properties of the canonicalizer. In particular, as shown in prior work the expectation of the branching program obtained from canonicalizing M is a close approximation to M.

Claim 3.2 ([SZ99, CDS22]). *For every $t \in \mathbb{N}$ and sub-stochastic matrix $M \in \mathbb{R}^{w \times w}$, we have*

$$\left\|\mathbb{E}\left[\mathtt{C_t}\left[M\right]\right] - M\right\| \leq w2^{-t}.$$

This extends to canonicalizing a sequence of substochastic matrices:

Claim 3.3. *For every $t \in \mathbb{N}$ and sub-stochastic matrices $M_1, \ldots, M_\ell \in \mathbb{R}^{w \times w}$, we have*

$$\left\|\mathbb{E}\left[\mathtt{C_t}\left[M_1, \ldots, M_\ell\right]\right] - M_1 \cdots M_\ell\right\| \leq w\ell 2^{-t}.$$

Proof. Deferred to Appendix A. □

We can further extend this to show "canonicalizations of similar matrices are similar".

Lemma 3.4. *Given sub-stochastic $M_1, \ldots, M_\ell$ and $\widetilde{M_1}, \ldots, \widetilde{M_\ell}$ in $\mathbb{R}^{w \times w}$, if $\|M_i - \widetilde{M_i}\| \leq \gamma$ for every i, then for every $t \in \mathbb{N}$,*

$$\left\|\mathbb{E}\left[\mathtt{C_t}\left[M_1, \ldots, M_\ell\right]\right] - \mathbb{E}\left[\mathtt{C_t}\left[\widetilde{M_1}, \ldots, \widetilde{M_\ell}\right]\right]\right\| \leq 2\ell w 2^{-t} + \ell\gamma.$$

Proof. Deferred to Appendix A. □

Note that this shows the expectations of the canonicalizations are close, but does not show the canonicalizations are close as labeled branching programs. Proving this stronger result is the primary contribution of the section.

3.1 Proof of Theorem 1.5

We now state the main lemma, which proves that canonicalizations of close matrices have similar structures as branching programs. In particular, the edges on which they differ are rare, and can be predicted in advance given only M.

Lemma 3.5. *Given $t \in \mathbb{N}$ and sub-stochastic $M_1, \ldots, M_\ell \in \mathbb{R}^{w \times w}$ and $\gamma \geq 2^{-t}$, there exists an ordered branching program E of width $w + 2$ and length ℓ and alphabet $[2^t]$ such that:*

1. *For every $j \in [w]$,*
$$\Pr\left[E\left[j, U_{[2^t]^\ell}\right] = v_{acc}\right] \leq 3w\ell \cdot \gamma.$$
2. *For every $\widetilde{M_1}, \ldots, \widetilde{M_\ell}$ satisfying $\left\|M_i - \widetilde{M_i}\right\| \leq \gamma$ for every i, we have that for every state $j \in [w]$ and every input $x \in [2^t]^\ell$,*
$$\mathtt{C_t}\left[M_1, \ldots, M_\ell\right][j, x] \neq \mathtt{C_t}\left[\widetilde{M_1}, \ldots, \widetilde{M_\ell}\right][j, x] \implies E[j, x] = v_{acc}.$$

Proof. For every $i \in [\ell]$ and state $j \in [w]$, let $s_k := \sum_{l=1}^{k} (M_i)_{j,l}$ for every $k \in [w]$. Define the boundary set $BD(M_i)_j \subset [2^t]$ of edges for state j in layer i as σ for which

$$BD(M_i)_j := \left\{\sigma : \sigma \cdot 2^{-t} \in \bigcup_{k=1}^{w} [s_k - \gamma, s_k + \gamma]\right\}.$$

Let E be the program (with $w+2$ states) that has identical states and transitions to $\mathtt{C_t}\left[M_1,\ldots,M_\ell\right]$ except it transitions to state v_{acc} if and only if the input ever traverses such an edge (and once it reaches v_{acc} always stays at this state). Observe that for every state j and every layer i, a random edge from j lies in the boundary set with low probability. In particular,

$$\Pr_{\sigma \leftarrow U_{[2^t]}}\left[\mathtt{C_t}\left[M_i\right][j,\sigma] \in BD(M_i)_j\right] \leq 2\gamma \cdot w + \frac{w}{2^t} \leq 3\gamma w$$

and so a union bound over steps proves (1).

Now fix $\widetilde{M}_1,\ldots,\widetilde{M}_\ell$ satisfying $\left\|M_i - \widetilde{M}_i\right\| \leq \gamma$ for every i. Observe that the boundary sets and error test program have been defined without reference to $\widetilde{M}$.

Claim 3.6. *For every $i \in [\ell]$ and $j \in [w]$, $BD(M_i)_j$ contains all edges σ on which $\mathtt{C_t}\left[M_i\right][j,\sigma] \neq \mathtt{C_t}\left[\widetilde{M}_i\right][j,\sigma]$.*

Proof. Fixing i, j, for each $k \in [w]$ let

$$\tilde{s}_k := \sum_{l=1}^{k} (\widetilde{M}_i)_{j,l},$$

and recall s_k is defined analogously. We have that for every k,

$$|s_k - \tilde{s}_k| = \left|\sum_{l=1}^{k}(\widetilde{M}_i)_{j,l} - \sum_{l=1}^{k}(M_i)_{j,l}\right| \leq \sum_{l=1}^{k}\left|(\widetilde{M}_i)_{j,l} - (M_i)_{j,l}\right| \leq \left\|\widetilde{M}_i - M_i\right\| \leq \gamma.$$

Thus for every k,

$$[s_k, \tilde{s}_k] \subset [s_k - \gamma, s_k + \gamma].$$

By definition of the canonicalizer, for every σ such that $\mathtt{C_t}\left[M_i\right][j,\sigma] \neq \mathtt{C_t}\left[\widetilde{M}_i\right][j,\sigma]$, we must have $\sigma \cdot 2^{-t} \in [s_k, \tilde{s}_k] \subset [s_k - \gamma, s_k + \gamma]$ for some k, and so $\sigma \in BD(M_i)_j$ as claimed. $\square$

Then we observe that for every j, x such that $\mathtt{C_t}\left[M_1,\ldots,M_\ell\right][j,x] \neq \mathtt{C_t}\left[\widetilde{M}_1,\ldots,\widetilde{M}_\ell\right][j,x]$, there is i such that $v := \mathtt{C_t}\left[M_1,\ldots,M_i\right][j,x_{1..i}] = \mathtt{C_t}\left[\widetilde{M}_1,\ldots,\widetilde{M}_i\right][j,x_{1..i}]$ and $\mathtt{C_t}\left[\widetilde{M}_{i+1}\right][v,x_{i+1}] \neq \mathtt{C_t}\left[M_{i+1}\right][v,x_{i+1}]$, since otherwise x would make identical transitions in both programs. But then the edge labeled x_{i+1} from v must have received different labels in $\mathtt{C_t}\left[M_{i+1}\right]$ and $\mathtt{C_t}\left[\widetilde{M}_{i+1}\right]$ and thus lie in $BD(M_{i+1})_j$ by Claim 3.6, and so by definition of E we must have $E[j,x] = v_{\mathrm{acc}}$. Note that it could have been the case that x transited a boundary edge before layer $i+1$, but this likewise causes $E[j,x]$ to reach state v_{acc} on input x. $\square$

We can then prove Theorem 1.5 using Lemma 3.5.

Theorem 1.5. *Fix $t \in \mathbb{N}$ and $\gamma \geq 2^{-t}$ and let $M_1,\ldots,M_\ell \in \mathbb{R}^{w\times w}$ be sub-stochastic matrices. Suppose $\mathtt{GEN} : \{0,1\}^d \to [2^t]^\ell$ is ε-good for $\mathtt{C_t}\left[M_1,\ldots,M_\ell\right]$ and E, where E is defined in Lemma 3.5 only in terms of $M_1,\ldots,M_\ell$ and t and γ. Then for every sub-stochastic $\widetilde{M}_1,\ldots,\widetilde{M}_\ell \in \mathbb{R}^{w\times w}$ where $\left\|M_i - \widetilde{M}_i\right\| \leq \gamma$ for every i, $\mathtt{GEN}$ is $\rho := 6w\ell\gamma + 2\varepsilon$-good for $\mathtt{C_t}\left[\widetilde{M}_1,\ldots,\widetilde{M}_\ell\right]$.*

Proof. By Lemma 3.5 applied with $\gamma = \gamma$ and $t = t$ we have

$$\begin{aligned}
&\left\|\overline{\mathtt{GEN}}\left[\mathtt{C_t}\left[M_1, \ldots, M_\ell\right]\right] - \overline{\mathtt{GEN}}\left[\mathtt{C_t}\left[\widetilde{M}_1, \ldots, \widetilde{M}_\ell\right]\right]\right\| \\
&\quad\le \max_j \left\{ \Pr_{x \leftarrow U_{\{0,1\}^d}}\left[\mathtt{C_t}\left[M_1, \ldots, M_\ell\right][j, \mathtt{GEN}(x)]\right] \neq \mathtt{C_t}\left[M_1, \ldots, M_\ell\right][j, \mathtt{GEN}(x)]\right] \right\} \\
&\quad\le \max_j \left\{ \Pr_{x \leftarrow U_{\{0,1\}^d}}\left[E[j, \mathtt{GEN}(x))] = v_{\mathrm{acc}}\right] \right\} && \text{(Property 2)} \\
&\quad\le 3w\ell \cdot \gamma + \varepsilon
\end{aligned}$$

where the final line follows from Property 1 and GEN being ε-good for E.

Furthermore, since by assumption GEN is ε-good for $\mathtt{C_t}\left[M_1, \ldots, M_\ell\right]$:

$$\left\|\overline{\mathtt{GEN}}\left[\mathtt{C_t}\left[M_1, \ldots, M_\ell\right]\right] - \overline{\mathbb{E}}\left[\mathtt{C_t}\left[M_1, \ldots, M_\ell\right]\right]\right\| \le \varepsilon$$

and by Lemma 3.4,

$$\left\|\overline{\mathbb{E}}\left[\mathtt{C_t}\left[M_1, \ldots, M_\ell\right]\right] - \overline{\mathbb{E}}\left[\mathtt{C_t}\left[\widetilde{M}_1, \ldots, \widetilde{M}_\ell\right]\right]\right\| \le 2\ell w 2^{-t} + \ell\gamma \le 3\ell w\gamma.$$

Thus, applying the triangle inequality we conclude that

$$\left\|\overline{\mathtt{GEN}}\left[\mathtt{C_t}\left[\widetilde{M}_1, \ldots, \widetilde{M}_\ell\right]\right] - \overline{\mathbb{E}}\left[\mathtt{C_t}\left[\widetilde{M}_1, \ldots, \widetilde{M}_\ell\right]\right]\right\| \le 6w\ell\gamma + 2\varepsilon. \qquad \square$$

We note that in the regime $w = n$, applying this result directly in the original framework of Saks and Zhou [SZ99] does not allow us to eliminate the random rounding step. This is because our error degrades with a factor of w per application, which could give a final error of $w^{\sqrt{\log n}} = n^{\omega(1)}$. However, we can use the approach of Cohen et al. to repair the loss at each level.

4 Proof of Theorem 1.4

We now apply Theorem 1.5 to prove Theorem 1.4. The analysis of our resulting algorithm is cleaner than prior approaches due to the absence of random rounding. In particular, we directly argue that the approximation at level i is close to the $2^{i \cdot \sqrt{\log n}}$th true power, rather than comparing to a shifted and rounded version of such. As we make the same parameter choices[3] as Cohen, Doron, and Sberlo (and the components of our algorithm are a strict subset of theirs), the space complexity follows essentially from their analysis, though we must be careful to avoid incurring an overhead of $O(\log n)$ bits per level for tracking indices or evaulating the sampler. We analyze the space complexity in Lemma 4.3.

We are now prepared to analyze the algorithm. Given n, w as specified in the upcoming theorem (where we may assume $n \ge w$ without loss of generality, as otherwise apply Theorem 1.9), we set parameters as follows:

$$r_1 = r_2 = \sqrt{\log n}, \quad t_2 = 20\log n, \quad t_1 = 2r_1 + \log w + 4, \quad \varepsilon = 2^{-r_1 - 4}, \quad \delta = 2n^{-5}, \quad R_1 = 2^{r_1}.$$

Let $\mathtt{NIS}_{h,\varepsilon,\delta}$ be the family of online-offline samplers from Lemma 2.4 with $n = R_1, w = w + 2, \Sigma = [2^{t_1}], \varepsilon = \varepsilon$ and $\delta = \delta$. We remark that the offline seed has length

$$m = O\left(r_1 \log(R_1 w 2^{t_1}/\varepsilon) + \log(1/\delta) + \log(1/\varepsilon)\right) = O\left(\log n + \sqrt{\log n} \cdot \log w\right).$$

[3]With the exception of δ, which we must take to be order $1/n$ to survive a bound over the test programs.

Furthermore, the online seed has length

$$d = O\left(\log(R_1 w 2^{t_1}/\varepsilon) + \log\log(1/\delta)\right) = O\left(\sqrt{\log n} + \log w\right).$$

We give the formal description of the algorithm in Algorithm 2.

Algorithm 2: $\texttt{SZ}(M_1, \ldots, M_n, h)$

```
1 Given $h \in \{0,1\}^m$, let $\texttt{NIS} := \texttt{NIS}_{h,\varepsilon,\delta}$ be the sampler with parameters set above.
2 return $\texttt{IMM}(\sqrt{\log n}, 0, \texttt{NIS})$.
```

Algorithm 3: $\texttt{IMM}(i, j, \texttt{NIS})$

```
1 if $i = 0$ then
2     return $M_j^0$;
3 end
4 for $l \in [j, j + R_1]$ do
5     Let $\widetilde{M}_l^{i-1} := \texttt{IMM}(i-1, R_1 \cdot j + l, \texttt{NIS})$.
6 end
7 Let
      $$\left\{\widetilde{M}_{k,l}\right\}_{k,l \in [R_1]} := \left\{\underset{k \to l}{\overline{\texttt{NIS}}}\left[\texttt{C}_{\texttt{t}_1}\left[\widetilde{M}_j^{i-1}, \ldots, \widetilde{M}_{j+R_1}^{i-1}\right]\right]\right\}_{k,l \in [R_1]}$$
8 Set
      $$\widetilde{M}_j^i := \texttt{R}\left(\left\{\widetilde{M}_{j+l}^{i-1}\right\}_{l \in [R_1]}, \left\{\widetilde{M}_{k,l}\right\}_{k,l \in [R_1]}, t_2\right)$$
9 return $\widetilde{M}_j^i$.
```

We now show Algorithm 2 works with high probability over the outer seed h.

Theorem 4.1. *Given $n, w \in \mathbb{N}$ with $w \le n$ and arbitrary sub-stochastic matrices $M_1, \ldots, M_n \in \mathbb{R}^{w \times w}$, Algorithm 2 returns with probability $1 - n^2\delta$ over the outer seed $h \leftarrow \{0,1\}^m$ a matrix $\widetilde{M} = \texttt{SZ}(M_1, \ldots, M_n, h)$ satisfying*

$$\left\| M_1 \cdots M_n - \widetilde{M} \right\| \le 1/n^3.$$

Moreover, $\texttt{SZ}(M_1, \ldots, M_n, h)$ runs in space

$$O\left(\left(\log n + \sqrt{\log n} \cdot \log(w)\right) \log\log(n)\right)$$

We first define the true powers that we wish to approximate, and their corresponding canonicalizations.

Definition 4.2. Given $n, w, t_1 \in \mathbb{N}$ and sub-stochastic $M_1^0, \ldots, M_n^0 \in \mathbb{R}^{w \times w}$, define

$$M_j^i = M_{j \cdot R_1}^{i-1} \cdot M_{j \cdot R_1 + 1}^{i-1} \cdots M_{j \cdot R_1 + R_1 - 1}^{i-1}$$

and let $\mathcal{P} = \left\{\texttt{C}_{\texttt{t}_1}\left[M_j^i, \ldots, M_{j+R_1}^i\right]\right\} \cup \mathcal{E}$ where $\mathcal{E}$ is the family of error testers of Lemma 3.5 applied with $t = t_1$ and $\gamma = \gamma$ (for γ to be globally chosen later) to every subproduct $M_j^i, \ldots, M_{j+R_1}^i$.

Note that we have $M_1^{r_2} = M_1^0 \cdots M_n^0$ by definition. We first prove the correctness, then analyze the space consumption.

Proof of Correctness of Theorem 4.1. We condition on the event that $\texttt{NIS} := \texttt{NIS}_{h,\varepsilon,\delta}$ is ε-good for the set of programs $\mathcal{P}$ as defined in Definition 4.2. This occurs with probability at least $1 - n^2\delta$ by Lemma 2.4 and the fact that there are at most $nw \leq n^2$ such programs. Subsequent to this assumption (which requires a union bound over nw bad events, rather than w), the proof does not change if we assume all base matrices are equal, so we do so for clarity. We maintain the following invariant at level i of the algorithm:

$$\left\|\widetilde{M}^i - M^i\right\| \leq \frac{2^{2i\cdot r_1}}{n^{10}}$$

Ensuring this invariant holds certainly suffices to complete the proof. Assuming the invariant holds for level i, we now verify that the conditions of Theorem 1.5 are satisfied for $t = t_1$ and $\ell = R_1$ and $\varepsilon = \varepsilon$ and $\gamma := 2^{-t_1}$ and $M = M^i$ and $\widetilde{M} = \widetilde{M}^i$. We have that $\texttt{NIS}$ is ε-good for M^i and the associated error tester by assumption. Furthermore by the invariant we have

$$\left\|\widetilde{M}^i - M^i\right\| \leq \frac{1}{n^2} \leq \gamma.$$

Therefore by Theorem 1.5 applied to the generator $\texttt{NIS}$, we obtain for every $j \in [R_1]$,

$$\left\|\overline{\mathbb{E}}\left[\texttt{C}_{\texttt{t}_1}\left[\widetilde{M}^i, \ldots, \widetilde{M}^i\right]\right] - \underset{1\to j}{\underline{\texttt{NIS}}}\left[\texttt{C}_{\texttt{t}_1}\left[\widetilde{M}^i, \ldots, \widetilde{M}^i\right]\right]\right\| \leq 6wR_1\gamma + 2\varepsilon \leq \frac{1}{10 \cdot R_1}$$

And thus by Claim 3.3 for every $j \in [R_1]$,

$$\left\|(\widetilde{M}^i)^j - \underset{1\to j}{\underline{\texttt{NIS}}}\left[\texttt{C}_{\texttt{t}_1}\left[\widetilde{M}^i, \ldots, \widetilde{M}^i\right]\right]\right\| \leq \frac{1}{10 \cdot R_1} + wR_1 2^{-t_1} \leq \frac{1}{5 \cdot R_1}.$$

Therefore, recalling

$$\widetilde{M}^{i+1} := \texttt{R}\left(\widetilde{M}^i, \left\{\underset{1\to j}{\underline{\texttt{NIS}}}\left[\texttt{C}_{\texttt{t}}\left[\widetilde{M}^i, \ldots, \widetilde{M}^i\right]\right]\right\}_j, t_2\right)$$

by Lemma 2.5 we have

$$\left\|\widetilde{M}^{i+1} - (\widetilde{M}^i)^{R_1}\right\| \leq 2 \cdot R_1 w^2 2^{-t_2} \leq \frac{1}{n^{10}}. \tag{3}$$

Thus,

$$\begin{aligned}
\left\|\widetilde{M}^{i+1} - M^{i+1}\right\| &\leq \left\|\widetilde{M}^{i+1} - (\widetilde{M}^i)^{R_1}\right\| + \left\|(\widetilde{M}^i)^{R_1} - M^{i+1}\right\| \\
&\leq \frac{1}{n^{10}} + \left\|(\widetilde{M}^i)^{R_1} - (M^i)^{R_1}\right\| && (3) \\
&\leq \frac{2}{n^{10}} + R_1 \cdot \left\|\widetilde{M}^i - M^i\right\| && \text{Claim 2.1} \\
&\leq \frac{2^{2(i+1)\cdot r_1}}{n^{10}}
\end{aligned}$$

which maintains the invariant for the next level.

Note that since that $i \leq \sqrt{\log n}$, this means that our error is bounded with

$$\frac{2^{2i\cdot r_1}}{n^{10}} \leq \frac{2^{2\sqrt{\log n}\cdot\sqrt{\log n}}}{n^{10}} \leq \frac{1}{n^8}.$$

□

Lemma 4.3. $\texttt{SZ}(M_1, \ldots, M_n, h)$ *runs in space*

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

Proof. We note that at no point do we explicitly write down the matrix $\widetilde{M_j^i}$, which would require $w^2 \log(nw)$ bits. Instead, whenever $\texttt{IMM}(i+1, j)$ requests a bit of $\widetilde{M_{j'}^i}$, we recurse on $\texttt{IMM}(i, j')$ and determine only this bit, then return control to level $i+1$. This process is formalized as the composition of space-bounded algorithms in Lemma A.2.

We look at the individual space complexities of the components of our algorithm:

1. First, we note that the seed length for the generator requires space (paid once)
$$O\left(\log n + \sqrt{\log n} \cdot \log w\right).$$

2. Each function $\texttt{IMM}(i, j)$ produces $w^2 t_2 = O(w^2 \log n)$ bits of output, and so by Lemma A.2 we require $O(\log w + \log\log n)$ bits per level to track the index of the bit to be output.

3. The online space for $\texttt{NIS}$ requires space (paid once per level)
$$O\left(\sqrt{\log n} + \log w\right).$$

4. By Lemma 2.4, we require space $O(\log n + \sqrt{\log n} \cdot \log w)$ to evaluate $\texttt{NIS}_h(x)$ on online input x, and as we do not touch the input or output during this time, this space can be reused between levels and so only needs to be paid for once.

5. Richardson iteration requires space (paid once per level)
$$O\left(\log^2 t_2 + \log t_2 \cdot \log(R_1 w t_1)\right) = O\left((\log\log n)^2 + \log\log n \cdot \left(\log w + \sqrt{\log n}\right)\right).$$

6. The canonicalizer requires space (paid once per level)
$$O(\log(t_1 t_2 w R_1)) = O(\sqrt{\log n} + \log w + \log\log n).$$

7. Specifying which matrices on the input tape should be multiplied can be done with $O(\log R_1) = O(\sqrt{\log n})$ bits per level of recursion.

To justify the last statement above, we see that if we assume the algorithm stores the index of which subproblem is currently being solved in each level, it can directly compute which matrices are supposed to be multiplied at any given step. Each index takes space $O(\log R_1)$, and is stored only once for each level in the call tree. Bringing this together, we note that the maximum recursion depth of our algorithm is $r_2 = \sqrt{\log n}$, and the space complexity of each recursive level is $O(\log w \log\log n + \sqrt{\log n} \log\log n)$. This means in total, the space required for the call tree is

$$O\left(\sqrt{\log n} \cdot \log w \cdot \log\log n + \log n \cdot \log\log n\right),$$

and this complexity does not change when we account for the offline seed of Nisan's generator. $\square$

Finally, we can use Theorem 4.1 to prove Theorem 1.4. We note that Cohen et al. [CDR+21] obtain a $1/n$ approximation by taking the median of each entry over the offline randomness (as their algorithm fails with probability $1/w \gg 1/n$ over h), but we obtain failure probability $1/n$ over h, so we take the average for simplicity.

Proof of Theorem 1.4. By our choice of δ, with probability at least $1 - 1/2n^3$ over the outer seed we obtain a final matrix $\widetilde{M}$ satisfying $\|\widetilde{M} - M_1 \cdots M_n\| \leq n^{-3}$. For a bad h, we receive a matrix with distance at most $w \leq n$ in ℓ_∞ distance, and so the theorem follows from letting $\mathcal{A}_1$ be the algorithm that returns the average of Algorithm 2 over h. □

By applying a final layer of Richardson iteration, we can obtain an arbitrary low-accuracy estimate at mild additional cost in space. We use a more precise statement of Richardson iteration, but we defer its statement to Section 6, where we use it as part of the inner loop of an algorithm.

Proof of Theorem 1.7. Let Algorithm $\mathcal{A}'_1$ be the algorithm that applies Lemma 6.1 to $M_1, \dots, M_n$ and $\{\widetilde{M}_{i,j}\}_{i,j\in[n]}$ with error ε, where for i, j, $\widetilde{M}_{i,j}$ is the output of Theorem 1.4 applied to $M_i, \dots, M_j$. The correctness is direct from the correctness of Theorem 1.4, and the space complexity follows from Lemma 6.1 and the composition of space bounded algorithms (Lemma A.2). □

5 Naive Saks-Zhou For Small Width

In this section we prove that the naive Saks-Zhou algorithm succeeds without random shifts as long as $w = O(2^{\sqrt{\log n}})$. We remark that this gives a space $O(\log^{3/2} n)$ algorithm for derandomizing width $\exp(\log^{1/2} n)$ branching programs whose only white-box step is repeatedly locally monotonizing subprograms (see Remark 1.6). For the remainder of the section, let

$$t = 20 \log n, \quad \varepsilon = n^{-10}, \quad \delta = n^{-6}, \quad r_1 = r_2 = \sqrt{\log n}, \quad R_1 = 2^{r_1}.$$

Let $\texttt{NIS}_{h,\varepsilon,\delta}$ be the family of online-offline samplers from Lemma 2.4 with $n = R_1, w = w + 2, \Sigma = [2^t], \varepsilon = \varepsilon$ and $\delta = \delta$. We remark that the offline seed has length

$$m = O\left(r_1 \log(r_1 w 2^t/\varepsilon) + \log(1/\delta) + \log(1/\varepsilon)\right) = O\left(\log^{3/2} n\right).$$

Furthermore, the online seed has length

$$d = O\left(\log(r_1 w 2^t/\varepsilon) + \log\log(1/\delta)\right) = O\left(\log n\right).$$

We now formally state this algorithm.

```
Algorithm 4: SZN($M_1, \dots, M_n, h$)
1 Given $h \in \{0,1\}^m$, let NIS := NIS$_{h,\varepsilon,\delta}$ be the sampler with parameters set above.
2 return NaiveIMM($\sqrt{\log n}$, 0, NIS).
```

We then state the theorem showing it is correct with high probability over the outer seed:

Theorem 5.1. *Given $n \in \mathbb{N}$ and $w \leq 2^{\sqrt{\log n}}$ and any sub-stochastic $M_1, \dots, M_n \in \mathbb{R}^{w\times w}$, Algorithm 4 returns with probability $1 - n^2\delta$ over the outer seed $h \leftarrow \{0,1\}^m$ a matrix $\widetilde{M} =$ $\texttt{SZN}(M_1, \dots, M_n, r_1, r_2, h)$ satisfying*

$$\left\|M_1 \cdots M_n - \widetilde{M}\right\| \leq 1/n^3.$$

Moreover, $\texttt{SZN}(M_1, \dots, M_n, h)$ *runs in space* $O\left(\log^{3/2} n\right)$.

Algorithm 5: Algorithm $\mathtt{NaiveIMM}(i, j, \mathtt{NIS})$

```
1 if $i = 0$ then
2     return $M_j^0$
3 end
4 for $l \in [j, j + R_1]$ do
5     let $\widetilde{M}_l^{i-1} := \mathtt{NaiveIMM}(i-1, R_1 \cdot j + l, \mathtt{NIS})$.
6 end
7 return $\overline{\underset{1\to R_1}{\mathtt{NIS}}}\left[\mathtt{C}_{\mathtt{t}_1}\left[\widetilde{M}_j^{i-1}, \ldots, \widetilde{M}_{j+R_1}^{i-1}\right]\right]$.
```

We construct the true powers identically to Definition 4.2 (with $t_1 = t$ and $\gamma_1, \ldots, \gamma_{r_2}$ to be chosen later, where the choice depends on the recursion level and nothing else). Furthermore, the space complexity directly follows from the analysis in the prior section, where we no longer pay for Richardson iterations. The remainder of the proof consists of setting parameters and walking through the same analysis.

Proof of Theorem 5.1. We condition on the event that $\mathtt{NIS} := \mathtt{NIS}_h$ is ε-good for $\mathcal{P}$ as defined in Definition 4.2, which occurs with probability at least $1 - n^2\delta$ as claimed. As in the prior case, subsequent to this assumption the proof does not change if we assume all base matrices are equal, so we do so for clarity. We maintain the following invariant at the start of the ith iteration of the loop:

$$\left\|\widetilde{M}^i - M^i\right\| \leq \frac{(8wR_1)^i R_1^i}{n^{10}}$$

Ensuring this invariant holds certainly suffices to complete the proof. We now verify that the conditions of Theorem 1.5 are satisfied for $t = t$ and $\varepsilon = \varepsilon$ and $\gamma_i := (8wR_1)^i R_1^i / n^{10}$ and $M = M^i$ and $\widetilde{M} = \widetilde{M}^i$. We have that $\mathtt{NIS}$ is ε-good for the relevant programs by assumption. Furthermore by the invariant we have

$$\left\|\widetilde{M}^i - M^i\right\| \leq \gamma_i.$$

Therefore by Theorem 1.5, recalling that

$$\widetilde{M}^{i+1} := \overline{\underset{1\to R_1}{\mathtt{NIS}}}\left[\mathtt{C}_{\mathtt{t}}\left[\widetilde{M}^i, \ldots, \widetilde{M}^i\right]\right]$$

we obtain

$$\left\|\mathbb{E}\left[\mathtt{C}_{\mathtt{t}}\left[\widetilde{M}^i, \ldots, \widetilde{M}^i\right]\right] - \widetilde{M}^{i+1}\right\| \leq 6wR_1\gamma_i + 2\varepsilon \leq (7wR_1)\gamma_i.$$

and thus by Claim 3.3,

$$\left\|(\widetilde{M}^i)^{R_1} - \widetilde{M}^{i+1}\right\| \leq (7wR_1)\gamma_i + wR_1 2^{-t} \leq (8wR_1)\gamma_i. \tag{4}$$

Thus,

$$\begin{aligned}
\left\|\widetilde{M}^{i+1} - M^{i+1}\right\| &\leq \left\|\widetilde{M}^{i+1} - (\widetilde{M}^i)^{R_1}\right\| + \left\|(\widetilde{M}^i)^{R_1} - M^{i+1}\right\| \\
&\leq (8wR_1)\gamma_i + \left\|(\widetilde{M}^i)^{R_1} - (M^i)^{R_1}\right\| && (4) \\
&\leq (8wR_1)\gamma_i + R_1 \cdot \|\widetilde{M}^i - M^i\| && \text{Claim 2.1} \\
&\leq \frac{(8wR_1)^{i+1}R_1^i}{n^{10}} + \frac{(8wR_1)^i R_1^{i+1}}{n^{10}} \\
&\leq \frac{(8wR_1)^{i+1}R_1^{i+1}}{n^{10}}.
\end{aligned}$$

which maintains the invariant for the next level. Note that since that $i \leq \sqrt{\log n}$, this means that our error is bounded with

$$\frac{(8wR_1)^i R_1^i}{n^{10}} \leq \frac{n^2 \cdot (8w)^{\sqrt{\log n}}}{n^{10}} = \frac{(8w)^{\sqrt{\log n}}}{n^8}. \qquad \square$$

Hence, we see that when $w \leq 2^{\sqrt{\log n}}$, the original algorithm of Saks and Zhou works without random shifting. The proof of Theorem 1.8 from Theorem 5.1 is exactly analogous to that of Theorem 1.4 from Theorem 4.1, so we omit it.

6 Saks-Zhou With Constant Richardson Iterations

To obtain space complexity matching Saks-Zhou without random rounding (with no loglog factors), we state a more precise version of Richardson iteration that utilizes stronger guarantees on the initial error:

Lemma 6.1 ([AKM+20, CDR+21, PV21, CDS22])**.** *There exists an algorithm* $\mathtt{R}$ *that, given* $t, k \in \mathbb{N}$ *and sub-stochastic* $M \in \mathbb{R}^{w \times w}$ *and* $\left(\widetilde{M}_i\right)_{i\in[n]}$ *and* $k \in \mathbb{N}$ *such that for all* i $\|M^i - \widetilde{M}_i\| \leq 1/(nw)^2$ *returns a substochastic matrix* $\mathtt{R}(M, (\widetilde{M}_i)_i, t, k)$ *where each entry is represented by at most* t *bits of precision satisfying*

$$\left\|\mathtt{R}\left(M, \left\{\widetilde{M}_i\right\}_i, t, k\right) - M^n\right\| \leq (nw)^{-k}.$$

Furthermore, $\mathtt{R}$ *runs in space* $O(\log^2 k + \log(k)\log(tnT))$ *where* T *is the maximum bit complexity of* M *and* $\{\widetilde{M}_i\}$*.*

Note that the above lemma implies that with only $O(1)$ Richardson iterations, we can reduce the error from $\frac{1}{w^c}$ back down to $\frac{1}{w^{c'}}$ for arbitrary constants $c < c'$, and this consumes $O(\log nw)$ space per level.

As we are now in the regime where we may assume $w \geq n$ without loss of generality, we appeal to identity 2 and assume we are computing the nth power of a single $w \times w$ stochastic matrix. For the remainder of the section let

$$t = 20\log(w), \quad \varepsilon = w^{-10}, \quad \delta = w^{-6}, \quad r_1 = r_2 = \sqrt{\log n}, \quad R_1 = 2^{r_1}.$$

Let $\mathtt{NIS}_{h,\varepsilon,\delta}$ be the family of online-offline samplers from Lemma 2.4 with $n = R_1, w = w+2, \Sigma = [2^t], \varepsilon = \varepsilon$ and $\delta = \delta$. We remark that the offline seed has length

$$m = O\left(r_1 \log(R_1 w 2^t/\varepsilon) + \log(1/\delta) + \log(1/\varepsilon)\right) = O\left(\log^{3/2} n + \sqrt{\log n} \cdot \log w\right).$$

Furthermore, the online seed has length

$$d = O\left(\log(R_1 w 2^t/\varepsilon) + \log\log(1/\delta)\right) = O\left(\log nw\right).$$

We formally describe the inner loop as Algorithm 6.

Algorithm 6: Algorithm $\mathtt{SZC}(M, h)$

```
1 Given $h \in \{0,1\}^m$, let $\mathtt{NIS}_{h,\varepsilon,\delta}$ be the sampler with parameters set above.
2 Set $\widetilde{M}^0 := M$.
3 for $i = 1, \ldots r_2$ do
4     Let
          $$\left\{\widetilde{M}_k\right\}_{k\in[R_1]} := \left\{\underset{1\to k}{\overline{\mathtt{NIS}}}\left[\mathtt{C}_{\mathtt{t}_1}\left[\widetilde{M}^{i-1}_1, \ldots, \widetilde{M}^{i-1}_{R_1}\right]\right]\right\}_{k\in[R_1]}.$$
5     Set
          $$\widetilde{M}^i := \mathtt{R}\left(\widetilde{M}^{i-1}, \left\{\widetilde{M}_k\right\}_{k\in[R_1]}, t, 10\right).$$
6 end
7 return $\widetilde{M}^{r_2}$
```

We then show Algorithm 6 is correct with high probability over the outer seed:

Theorem 6.2. *Given $n, w \in \mathbb{N}$ with $w \geq n$ and any stochastic $M \in \mathbb{R}^{w\times w}$, Algorithm 6 returns with probability $1 - w^2\delta$ over the outer seed $h \leftarrow \{0,1\}^m$ a matrix $\widetilde{M} = \mathtt{SZC}(M, h)$ satisfying*

$$\left\|M^n - \widetilde{M}\right\| \leq 1/w^3.$$

Moreover, $\mathtt{SZC}(M, h)$ runs in space

$$O\left(\log^{3/2} n + \sqrt{\log n} \cdot \log w\right).$$

We construct the true powers identically to Definition 4.2 (with $t_1 = t$). The rest of the proof is analogous to Theorem 4.1, except that we set γ to be $1/w^c$ and so require only a constant number of Richardson iterations per level. The space complexity follows from Lemma 6.1, as we now require $O(\log nw)$ space per level for the Richardson iterations (and the online seed).

7 Acknowledgements

We thank Salil Vadhan and David Zuckerman for their insightful comments during the writing of this paper.

References

[AKM+20] AmirMahdi Ahmadinejad, Jonathan A. Kelner, Jack Murtagh, John Peebles, Aaron Sidford, and Salil P. Vadhan. High-precision estimation of random walks in small space. In *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1295–1306, 2020.

[Arm98] Roy Armoni. On the derandomization of space-bounded computations. In *Randomization and approximation techniques in computer science (Barcelona, 1998)*, volume 1518 of *Lecture Notes in Comput. Sci.*, pages 47–59. Springer, Berlin, 1998.

[BCG20] Mark Braverman, Gil Cohen, and Sumegha Garg. Pseudorandom pseudo-distributions with near-optimal error for read-once branching programs. *SIAM J. Comput.*, 49(5), 2020.

[BGZ21] Mark Braverman, Sumegha Garg, and Or Zamir. Tight space complexity of the coin problem. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1068–1079. IEEE, 2021.

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

[CDR$^+$21] Gil Cohen, Dean Doron, Oren Renard, Ori Sberlo, and Amnon Ta-Shma. Error Reduction for Weighted PRGs Against Read Once Branching Programs. In *Proceedings of the 36th Computational Complexity Conference (CCC)*, pages 22:1–22:17, 2021.

[CDS22] Gil Cohen, Dean Doron, and Ori Sberlo. Approximating large powers of stochastic matrices in small space. *Electron. Colloquium Comput. Complex.*, TR22-008, 2022.

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

[CHRT18] Eshan Chattopadhyay, Pooya Hatami, Omer Reingold, and Avishay Tal. Improved pseudorandomness for unordered branching programs through local monotonicity. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 363–375. ACM, 2018.

[CL20] Eshan Chattopadhyay and Jyun-Jie Liao. Optimal error pseudodistributions for read-once branching programs. In *Proceedings of the 35th Computational Complexity Conference (CCC)*, pages 25:1–25:27, 2020.

[FK18] Michael A. Forbes and Zander Kelley. Pseudorandom generators for read-once branching programs, in any order. In *Proceedings of the 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 946–955, 2018.

[Hoz21] William M. Hoza. Better pseudodistributions and derandomization for space-bounded computation. In *Proceedings of the 25th International Conference on Randomization and Computation (RANDOM)*, pages 28:1–28:23, 2021.

[HZ20] William M. Hoza and David Zuckerman. Simple optimal hitting sets for small-success RL. *SIAM J. Comput.*, 49(4):811–820, 2020.

[INW94] Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 356–364. ACM, 1994.

[MRSV17] Jack Murtagh, Omer Reingold, Aaron Sidford, and Salil P. Vadhan. Derandomization beyond connectivity: Undirected laplacian systems in nearly logarithmic space. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 801–812, 2017.

[MRSV19] Jack Murtagh, Omer Reingold, Aaron Sidford, and Salil Vadhan. Deterministic approximation of random walks in small space. In Dimitris Achlioptas and László A. Végh, editors, *Proceedings of the 23rd International Conference on Randomization and Computation (RANDOM '19)*, volume 145 of *LIPIcs*, 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 *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 626–637. ACM, 2019.

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

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

[PV21] Edward Pyne and Salil Vadhan. Pseudodistributions That Beat All Pseudorandom Generators (Extended Abstract). In *Proceedings of the 36th Annual Computational Complexity Conference (CCC)*, pages 33:1–33:15, 2021.

[RVW02] Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders. *Annals of Mathematics*, 155(1), January 2002.

[RVW04] Omer Reingold, Salil Vadhan, and Avi Wigderson. A note on extracting randomness from santha–vazirani sources. Unpublished manuscript, September 2004.

[Sav70] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4:177–192, 1970.

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

A Deferred Proofs

We collect proofs of claims regarding the accuracy of various truncation and rounding procedures. In all cases, our results are insensitive to polynomial losses in the length and width of the relevant subprograms, and constant factors in terms of the bit complexity (i.e. the parameters t_1, t_2).

We first extend Claim 3.2 to canonicalizations of sequences of matrices.

Claim 3.3. *For every $t \in \mathbb{N}$ and sub-stochastic matrices $M_1, \ldots, M_\ell \in \mathbb{R}^{w \times w}$, we have*

$$\left\| \mathbb{E} \left[\mathtt{C_t} \left[M_1, \ldots, M_\ell \right] \right] - M_1 \cdots M_\ell \right\| \leq w\ell 2^{-t}.$$

Proof. We have

$$\begin{aligned}\left\|\overline{\mathbb{E}}\left[\mathtt{C_t}\left[M_1,\dots,M_\ell\right]\right]-M_1\cdots M_\ell\right\| &= \left\|\overline{\mathbb{E}}\left[\mathtt{C_t}\left[M_1\right]\right]\cdots\overline{\mathbb{E}}\left[\mathtt{C_t}\left[M_\ell\right]\right]-M_1\cdots M_\ell\right\|\\ &\le \ell w 2^{-t}\end{aligned}$$

where the second step uses Claim 2.1 with $A_i=\overline{\mathbb{E}}\left[\mathtt{C_t}\left[M_i\right]\right]$ and $B_i=M_i$ and $\delta=w2^{-t}$ from Claim 3.2. □

We observe that this easily implies that the expectations of canonicalizations of similar matrices are similar.

Lemma 3.4. *Given sub-stochastic $M_1,\dots,M_\ell$ and $\widetilde{M_1},\dots,\widetilde{M_\ell}$ in $\mathbb{R}^{w\times w}$, if $\|M_i-\widetilde{M_i}\|\le\gamma$ for every i, then for every $t\in\mathbb{N}$,*

$$\left\|\mathbb{E}\left[\mathtt{C_t}\left[M_1,\dots,M_\ell\right]\right]-\mathbb{E}\left[\mathtt{C_t}\left[\widetilde{M_1},\dots,\widetilde{M_\ell}\right]\right]\right\|\le 2\ell w2^{-t}+\ell\gamma.$$

Proof. We have

$$\begin{aligned}&\left\|\overline{\mathbb{E}}\left[\mathtt{C_t}\left[M_1,\dots,M_\ell\right]\right]-\overline{\mathbb{E}}\left[\mathtt{C_t}\left[\widetilde{M_1},\dots,\widetilde{M_\ell}\right]\right]\right\|\\ &\quad\le \left\|\overline{\mathbb{E}}\left[\mathtt{C_t}\left[M_1,\dots,M_\ell\right]\right]-M_1\cdots M_\ell\right\|+\left\|M_1\cdots M_\ell-\widetilde{M_1}\cdots\widetilde{M_\ell}\right\|+\left\|\widetilde{M_1}\cdots\widetilde{M_\ell}-\overline{\mathbb{E}}\left[\mathtt{C_t}\left[\widetilde{M_1},\dots,\widetilde{M_\ell}\right]\right]\right\|\\ &\quad\le 2w\ell2^{-t}+\ell\gamma\end{aligned}$$

where the final line uses Claim 3.3 and Claim 2.1. □

We recall the formal statement of the composition of space-bounded algorithms:

Lemma A.1 ([CDS22])**.** *Let $f_1,f_2:\{0,1\}^*\to\{0,1\}^*$ be computable in space $s_1,s_2:\mathbb{N}\to\mathbb{N}$, where $s_1(n),s_2(n)\ge\log n$. Then, $f_1\circ f_2(x)$ can be computed in space*

$$O(s_1(\ell_2(n))+s_2(n)),$$

where $\ell_2(n)$ is a bound on the length of the output of $f_2(x)$ on inputs of length n.

We can apply this lemma to the case of a single function being composed with itself many times:

Lemma A.2 ([CDS22])**.** *Let $f:\{0,1\}^*\to\{0,1\}^*$ be computable in space $s:\mathbb{N}\to\mathbb{N}$, where $s(n)\ge\log n$. Then, $g(x,k)=f\circ f\circ\cdots\circ f(x)$ can be computed in space*

$$O\left(\sum_{i=0}^{k-1}s(\ell_i(n))\right),$$

where $\ell_i(n)$ is a bound on the length of the output of $g(x,i)$ on inputs of length n.

A.1 Samplers With Low Failure Probability

We prove that there exists a generator with our required properties. We first recall the statement:

Theorem 2.3 ([Nis92, RVW02, CL20]). *Given $n, w, |\Sigma| \in \mathbb{N}$ and $\varepsilon, \delta > 0$, there exists a generator $\mathtt{NIS} : \{0,1\}^m \times \{0,1\}^d \to \Sigma^n$ such that for every length n, width w branching program B we have:*

$$\Pr_{h \leftarrow U_m}\left[\left\|\overline{\mathtt{NIS}(h,\cdot)}[B] - \overline{\mathbb{E}}[B]\right\| \leq \varepsilon\right] \geq 1-\delta$$

and $m = O(\log(n)\log(nw|\Sigma|/\varepsilon) + \log(n/\delta))$ and $d = O(\log(nw|\Sigma|/\varepsilon) + \log\log(n/\delta))$. Equivalently, for every branching program B, with probability $1-\delta$ over h $\mathtt{NIS}_h := \mathtt{NIS}(h,\cdot)$ is ε-good for B. Furthermore, $\mathtt{NIS}_h(x)$ can be evaluated in space $O(m)$ given two-way read-only access to the offline seed h.

We construct this in a standard fashion via the sampler trick. First, we recall both the original Nisan PRG, and a space-efficient averaging sampler.

Theorem A.3 ([Nis92]). *Given $n, w, |\Sigma| \in \mathbb{N}$ and $\varepsilon > 0$, there is a function $\mathtt{GEN} : \{0,1\}^s \to \Sigma^n$ with $m = O(\log n \log(nw|\Sigma|/\varepsilon)$ that can be evaluated in space $O(s)$ such that for every ordered branching program $B : [w] \times \Sigma^n \to [w]$ of length n and width w,*

$$\left\|\overline{\mathtt{GEN}}[B] - \overline{\mathbb{E}}[B]\right\| \leq \varepsilon.$$

Theorem A.4 ([RVW04, CL20]). *For every $\varepsilon, \delta > 0$ and $s \in \mathbb{N}$, there exists an averaging sampler $f : \{0,1\}^m \times \{0,1\}^d \to \{0,1\}^s$ such that $d = O(\log(1/\varepsilon) + \log\log(1/\delta))$ and $m = s + O(\log(1/\delta) + \log(1/\varepsilon))$. Formally, for every function $g : \{0,1\}^s \to \{0,1\}$, we have*

$$\Pr_{h \leftarrow U_m}\left[\left|\mathop{\mathbb{E}}_{x \leftarrow U_d}[g(f(h,x)] - \mathop{\mathbb{E}}_{y \leftarrow U_s}[g(y)]\right| \leq \varepsilon\right] \geq 1-\delta.$$

By combining these two ingredients we can construct the generator.

Proof of Theorem 2.3. Let $\mathtt{GEN} : \{0,1\}^s \to \Sigma^n$ be the function of Theorem A.3 with $n = n, w = w, |\Sigma| = |\Sigma|$ and $\varepsilon = \varepsilon/2$. Observe that for every ordered branching program B of length n and width w, we have

$$\left\|\overline{\mathbb{E}}[B \circ \mathtt{GEN}] - \overline{\mathbb{E}}[B]\right\| \leq \varepsilon/2.$$

Now let $f : \{0,1\}^m \times \{0,1\}^d \to \{0,1\}^s$ be the function of Theorem A.4 with $\delta = \delta/w, \varepsilon = \varepsilon/2w$. For every function $B \circ \mathtt{GEN}$ (where we choose a distinguished start and accept vertex and later take a union bound over ε and δ) we have

$$\Pr_{h \leftarrow U_m}\left[\left|\mathop{\mathbb{E}}_{x \leftarrow U_d}[B \circ \mathtt{GEN}(f(h,x))] - \mathop{\mathbb{E}}_{y \leftarrow U_s}[B \circ \mathtt{GEN}(y)]\right| \leq \varepsilon/2w\right] \geq 1-\delta/w.$$

and thus

$$\Pr_{h \leftarrow U_m}\left[\left\|\overline{\mathtt{GEN} \circ f(h,\cdot)}[B] - \overline{\mathbb{E}}[B]\right\| \leq \varepsilon/2 + \varepsilon/2\right] \geq 1-\delta$$

so letting $\mathtt{NIS}(h,\cdot) = \mathtt{GEN} \circ f(h,\cdot)$ we obtain the desired generator. $\square$

B Bounds Against Single Transition Branching Programs

To illustrate the added power obtained from derandomizing products $M_1 \cdots M_n$ of stochastic matrices $M_1, \ldots, M_n \in \mathbb{R}^{w \times w}$ versus derandomizing powers M^n of a stochastic matrix $M \in \mathbb{R}^{w \times w}$, we observe that the latter model corresponds to derandomizing ordered branching programs with a single fixed transition function for every layer. We show via a short combinatorial argument that this limitation can be severe:

Lemma B.1. *For every even n, the function $f : \{0,1\}^n \to \{0,1\}$ given by $f(x) = x_{n/2}$ cannot be computed by a single-transition ordered branching program of width $n/2 - 1$.*

We remark that f (and any parity on any subset of variables) can easily be computed by an ordered branching program (with distinct transition functions) of width 2.

Proof. Let B be a single-transition branching program of width w computing f, and let v_0 be the start state and $A \subset [w]$ be the set of states marked as accept in the final layer.

Claim B.2. *For arbitrary $u, v \in [w]$, if there exists $\sigma \in \{0,1\}^*$ such that $B[v, \sigma] = u$, there exists $\tau \in \{0,1\}^{\leq w}$ such that $B[v, \tau] = u$.*

Proof. This follows from the fact that if an $s - t$ path exists in a directed graph of size w, there must be an $s - t$ path of length at most w. ☐

Now assume for contradiction that $w \leq n/2 - 1$. Let $u := B[v_0, 0^{n/2-1}1]$. We first claim that for any $\sigma \in \{0,1\}^*$, $B[u, \sigma] \in A$. This follows from Claim B.2 and the fact that B must accept all strings of the form $B[v_0, 0^{n/2-1}1\sigma]$. But then there is $\tau \in \{0,1\}^{\leq w}$ such that $B[v_0, \tau] = u$, and hence $B[v_0, \tau 0^{n-|\tau|}] \in A$, and as $n - |\tau| > n/2$ this is a contradiction to the fact that B computes f. ☐