Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 63 (2018)

Simple Optimal Hitting Sets for Small-Success **RL**

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

David Zuckerman†
Department of Computer Science,
University of Texas at Austin
diz@cs.utexas.edu

April 9, 2018

Abstract

We give a simple explicit hitting set generator for read-once branching programs of width w and length r with known variable order. Our generator has seed length

$$O\left(\frac{\log(wr)\log r}{\max\{1, \log\log w - \log\log r\}} + \log(1/\varepsilon)\right). \tag{1}$$

This seed length improves on recent work by Braverman, Cohen, and Garg [BCG18]. In addition, our generator and its analysis are dramatically simpler than the work by Braverman et al. [BCG18]. Our generator's seed length improves on all the classic generators for space-bounded computation [Nis92, INW94, NZ96] when ε is small.

When $r \leq \operatorname{polylog} w$, our generator has optimal seed length $O(\log w + \log(1/\varepsilon))$. As a corollary, we show that every **RL** algorithm that uses r random bits can be simulated by an **NL** algorithm that uses only $O(r/\log^c n)$ nondeterministic bits, where c is an arbitrarily large constant. Finally, we show that any **RL** algorithm with small success probability ε can be simulated deterministically in space $O(\log^{3/2} n + \log n \log\log(1/\varepsilon))$. This improves on work by Saks and Zhou [SZ99], who gave an algorithm that runs in space $O(\log^{3/2} n + \sqrt{\log n}\log(1/\varepsilon))$.

1 Introduction

1.1 The power of randomness for space-bounded algorithms

A fundamental goal of complexity theory is to understand the extent to which randomness is useful for computation. After decades of research, it is widely conjectured that randomized decision algorithms can always be made deterministic with only a polynomial factor slowdown ($\mathbf{P} = \mathbf{BPP}$) and only a constant factor space blowup ($\mathbf{L} = \mathbf{BPL}$). In this paper, we focus on derandomizing **RL**, the class of languages decidable by randomized log-space algorithms with one-sided error.

After an n-bit input to a randomized log-space algorithm has been fixed, the behavior of the algorithm is described by a *read-once branching program* (ROBP). An ROBP of width w and length r is a layered digraph with $r+1$ layers and w vertices per layer. Each vertex not in the last layer has two outgoing edges to the next layer, labeled 0 and 1. The program P starts at a designated start vertex in the first layer, reads an r-bit string from left to right in the natural way, and either accepts or rejects depending on what vertex it arrives at in the final layer. This defines a function $P : \{0,1\}^r \to \{0,1\}$.

*Supported by the NSF GRFP under Grant DGE-1610403 and by a Harrington Fellowship from UT Austin.
†Supported by NSF Grant CCF-1526952, NSF Grant CCF-1705028, and a Simons Investigator Award (#409864).

ISSN 1433-8092

A natural approach to derandomizing **BPL** is to design an efficient *pseudorandom generator* (PRG). An ε-PRG for width-w, length-r ROBPs is a function $\mathsf{Gen} : \{0,1\}^s \to \{0,1\}^r$ such that for any such ROBP P,[1]

$$|\Pr[P(U_r) = 1] - \Pr[P(\mathsf{Gen}(U_s)) = 1]| \leq \varepsilon. \tag{2}$$

A *hitting set generator* (HSG) is a relaxation of a PRG, still suitable for derandomizing **RL**, where Eq. (2) is replaced with

$$\Pr[P(U_r) = 1] \geq \varepsilon \implies \exists x, P(\mathsf{Gen}(x)) = 1. \tag{3}$$

1.2 Previous generators

We will describe only some of the myriad generators that researchers have developed for ROBPs [AKS87, BNS92, Nis92, INW94, NZ96, Arm98, KNW08, ŠŽ11, GMR+12, BDVY13, GR14, BCG18]. Perhaps the best known generator is Nisan's PRG [Nis92], which has seed length

$$O(\log(wr/\varepsilon) \log r). \tag{4}$$

Better generators are known when $r \ll w$, which corresponds to derandomizing algorithms that only make a few coin tosses. For $r \leq O(\log^2 w / \log\log w)$ and $\varepsilon \geq 1/\operatorname{poly}(w)$, Ajtai, Komlós, and Szemerédi gave an HSG with optimal seed length $O(\log w)$ [AKS87].[2] For $r \leq \operatorname{polylog} w$ and $\varepsilon \geq 2^{-\log^{1-\Omega(1)} w}$, Nisan and Zuckerman gave a PRG with optimal seed length $O(\log w)$ [NZ96]. Armoni [Arm98] showed how to interpolate between Nisan's generator [Nis92] and the Nisan-Zuckerman generator [NZ96]. By improving an extractor in Armoni's construction, Kane et al. showed [KNW08] that for $r \geq \log w$, Armoni's PRG can be implemented to have seed length

$$O\left(\frac{\log(wr/\varepsilon) \log r}{\max\{1, \log\log w - \log\log(r/\varepsilon)\}}\right). \tag{5}$$

An ROBP that arises from a uniform randomized algorithm always satisfies $w \geq r$, but the ROBP model is still interesting even when $w \ll r$. Bogdanov et al. gave a PRG for $w = 2$ with optimal seed length $O(\log(r/\varepsilon))$ [BDVY13]. Šíma and Žák gave a 0.961-HSG for $w \leq 3$ with seed length $O(\log r)$ [ŠŽ11], and Gopalan et al. gave an HSG for $w \leq 3$ with seed length $\widetilde{O}(\log(r/\varepsilon))$ [GMR+12].

When $r = w$, Armoni's PRG is no better than Nisan's. Indeed, for the quarter century after Nisan announced his generator [Nis92], there was no improvement whatsoever for the case $r = w$. In an exciting recent development, Braverman, Cohen, and Garg [BCG18] gave an HSG with seed length

$$\widetilde{O}(\log(wr) \log r + \log(1/\varepsilon)). \tag{6}$$

Equation (6) improves on all prior generators when, e.g., $r = w$ and $\varepsilon = w^{-\log w}$. Unfortunately, as Braverman et al. recognize in their 67-page paper, their construction is "fairly involved" and their analysis "requires a significant amount of work" [BCG18].

1.3 Our results

1.3.1 A new HSG

In this paper, we explicitly construct a new HSG for ROBPs with seed length

$$O\left(\frac{\log(wr) \log r}{\max\{1, \log\log w - \log\log r\}} + \log(1/\varepsilon)\right). \tag{7}$$

[1] U_n is the uniform distribution on $\{0,1\}^n$.

[2] Ajtai et al.'s generator [AKS87] was not entirely subsumed by any subsequent papers until this one!

This seed length improves on all the classic generators [AKS87, BNS92, Nis92, INW94, NZ96, Arm98] and the recent generator by Braverman et al. [BCG18]. Our generator's seed length has optimal dependence on ε; this is the improvement over Eq. (5). When $r \leq \operatorname{polylog} w$, our generator has optimal seed length $O(\log w + \log(1/\varepsilon))$. When r is large, our generator is still interesting; for example, when $r = w$, our generator is the first with seed length $O(\log^2 w)$ for sub-polynomial values of ε such as $\varepsilon = w^{-\log w}$. Our construction and analysis are very simple.

One respect in which the generator by Braverman et al. [BCG18] is superior to ours is that their construction gives a *pseudorandom pseudodistribution* (PRPD). The notion of a PRPD, introduced by Braverman et al. [BCG18], is intermediate between an HSG and a PRG and is suitable for derandomizing **BPL**. Our construction does not give a PRPD.

1.3.2 Randomness vs. nondeterminism

An HSG can be thought of as stretching a short nondeterministic seed to a long pseudorandom string. Using our HSG, we show that every **RL** algorithm that uses r random bits can be simulated by an **NL** algorithm that uses only $O(r/\log^c n)$ nondeterministic bits, where c is an arbitrarily large constant. In other words, for log-space algorithms, nondeterministic bits are worth at least $\operatorname{polylog}(n)$ random bits apiece.

For comparison, the work of Ajtai et al. [AKS87] implies a simulation with $O\left(\frac{r}{\log n/\log\log n}\right)$ nondeterministic bits. The work of Nisan and Zuckerman [NZ96] implies the incomparable statement that for $r \leq 2^{\log^{1-\Omega(1)} n}$, every **RL** algorithm that uses r random bits can be simulated by an **RL** algorithm that uses only $O(r/\log^c n)$ random bits.

1.3.3 Derandomizing small-success algorithms

The standard definition of **RL** requires that given an input in the language, the algorithm should accept with probability at least $1/2$. We can more generally consider algorithms that merely accept with probability at least ε. Saks and Zhou showed [SZ99] that such languages can be decided deterministically in space $O(\log^{3/2} n + \sqrt{\log n}\log(1/\varepsilon))$. Using the same technique we use to analyze our HSG, we give a deterministic algorithm for such languages that runs in space $O(\log^{3/2} n + \log n \log\log(1/\varepsilon))$. For example, the Saks-Zhou algorithm only runs in space $O(\log^{3/2} n)$ if $\varepsilon \geq 1/\operatorname{poly}(n)$, whereas our algorithm runs in space $O(\log^{3/2} n)$ even when $\varepsilon = 2^{-2^{\Theta(\sqrt{\log n})}}$. In the extreme limit $\varepsilon = 2^{-\operatorname{poly}(n)}$, our algorithm recovers Savitch's theorem $\mathbf{NL} \subseteq \mathbf{DSPACE}(\log^2 n)$ [Sav70]; indeed, our algorithm relies on Savitch's algorithm [Sav70].

1.4 Techniques

Our results are based on an elementary structural lemma for ROBPs (Lemma 1). Roughly, the lemma says that from any vertex v, there's at least a $1/\operatorname{poly}(r)$ chance of reaching a set $\Lambda(v)$ of vertices such that reaching $\Lambda(v)$ represents making a lot of progress toward eventually accepting. The interesting case is $\varepsilon \ll 1/\operatorname{poly}(r)$.

Based on this lemma, we show how to convert any $(1/\operatorname{poly}(r))$-PRG for ROBPs into an ε-HSG. To explain our construction, for simplicity, consider $r = w$ and $\varepsilon = w^{-\log w}$. Our HSG uses a "hitter," a randomized algorithm that produces a list of $\operatorname{poly}(w)$ seeds to the given PRG. Our HSG selects $O(\log w)$ seeds from this list and outputs the concatenation of the corresponding pseudorandom strings. This works, because if the hitter does its job and our HSG selects appropriate seeds, then the first pseudorandom string leads from the start vertex, v_0, to a vertex $v_1 \in \Lambda(v_0)$. The second

pseudorandom string leads from v_1 to some $v_2 \in \Lambda(v_1)$. Then we go to $v_3 \in \Lambda(v_2)$, etc., and eventually accept.

Suppose we plug in Nisan's generator [Nis92] as the $(1/\operatorname{poly}(r))$-PRG in this construction. It has seed length $O(\log^2 w)$, so a high-quality hitter only needs $O(\log^2 w)$ random bits to produce its list. Our HSG needs an additional $O(\log^2 w)$ nondeterministic bits to select $O(\log w)$ seeds from the list. Finally, our HSG needs another $O(\log^2 w)$ nondeterministic bits to guess the distances from v_0 to $\Lambda(v_0)$, from v_1 to $\Lambda(v_1)$, etc. Thus, in total, our ε-HSG's seed length is only $O(\log^2 w)$.

In general, if the given $(1/\operatorname{poly}(r))$-PRG has seed length m, our ε-HSG has seed length $O(m + \log(wr/\varepsilon))$. To get the best seed length, we plug in Armoni's PRG [Arm98, KNW08].

2 Our generator

2.1 Construction

A (θ, δ)-*hitter* [Gol11] is a function $\mathsf{Hit} : \{0,1\}^\ell \times \{0,1\}^d \to \{0,1\}^m$ such that for any set $E \subseteq \{0,1\}^m$,

$$\Pr[U_m \in E] \geq \theta \implies \Pr_x[\exists y, \mathsf{Hit}(x,y) \in E] \geq 1 - \delta. \tag{8}$$

(This is equivalent to the notion of a *disperser*.) Our ε-HSG for width-w, length-r ROBPs is built from two ingredients:

- A $(\frac{1}{r^2})$-PRG for width-w, length-r ROBPs $\mathsf{BaseGen} : \{0,1\}^m \to \{0,1\}^r$.
- A $(\frac{1}{r^2}, \frac{1}{2wr})$-hitter $\mathsf{Hit} : \{0,1\}^\ell \times \{0,1\}^d \to \{0,1\}^m$.

A seed to our generator consists of a string $x \in \{0,1\}^\ell$, a positive integer $t \in \left\{1, 2, \ldots, \left\lceil \frac{\log(1/\varepsilon)}{\log(r/4)} \right\rceil + 1\right\}$, positive integers $r_1, \ldots, r_t$ with $r_1 + \cdots + r_t = r$, and strings $y_1, \ldots, y_t \in \{0,1\}^d$. The output of our generator is

$$\mathsf{Gen}(x, t, r_1, \ldots, r_t, y_1, \ldots, y_t) = \mathsf{BaseGen}(\mathsf{Hit}(x, y_1))|_{r_1} \circ \cdots \circ \mathsf{BaseGen}(\mathsf{Hit}(x, y_t))|_{r_t}. \tag{9}$$

Here, $\circ$ denotes string concatenation and $z|_{r_i}$ denotes the truncation of z to the first r_i bits. Our construction draws inspiration from the Nisan-Zuckerman generator [NZ96].

2.2 Correctness

Let P be a width-w, length-r ROBP with layers $L_0, L_1, \ldots, L_r$. Suppose $i \leq j$. If $u \in L_i, v \in L_j$, let $p(u,v)$ be the probability of landing at v when starting at u and reading U_{j-i}. If $U \subseteq L_i, V \subseteq L_j$, let

$$p(U,V) = \min_{u \in U} \sum_{v \in V} p(u,v). \tag{10}$$

Let $v_* \in L_r$ be the accepting vertex of P, which we may assume is unique without loss of generality. We now prove the structural lemma outlined in Section 1.4. Our lemma bears some resemblance to a lemma by Ajtai et al. [AKS87, Lemma 1].

Lemma 1. *Assume $r > 4$. Suppose $v \in L_i$, $i < r$. There is a positive integer $h(v)$ and a set $\Lambda(v) \subseteq L_{i+h(v)}$ so that*

1. *$p(v, \Lambda(v)) \geq \frac{2}{r^2}$, and*
2. *$\Lambda(v) = \{v_*\}$ or $p(\Lambda(v), v_*) \geq p(v, v_*) \cdot (r/4)$.*

Proof. Let $\alpha = p(v, v_*)$. If $\alpha \geq 2/r$, we can just let $\Lambda(v) = \{v_*\}$ and $h(v) = r - i$, so assume $\alpha < 2/r$. For $j > i$, define

$$\Lambda_j = \{u \in L_j : \alpha \cdot (r/4) \leq p(u, v_*) \leq \alpha \cdot (r/2)\}. \tag{11}$$

Say that Λ_j is *unlikely* if $p(v, \Lambda_j) < \frac{2}{r^2}$. Consider starting at v and reading uniform randomness. If Λ_j is unlikely, then the probability of passing through Λ_j and then ultimately accepting is given by

$$\sum_{u \in \Lambda_j} p(v, u) \cdot p(u, v_*) \leq p(v, \Lambda_j) \cdot \alpha \cdot (r/2) < \frac{\alpha}{r}. \tag{12}$$

There are at most r unlikely sets Λ_j. Therefore, by the union bound, the probability of passing through *some* unlikely Λ_j and then ultimately accepting is strictly less than α. So there is some path $v = u_i, u_{i+1}, \ldots, u_r = v_*$ that does not pass through any unlikely Λ_j.

Since u_{j+1} is an outneighbor of u_j,

$$p(u_j, v_*) \geq p(u_j, u_{j+1}) \cdot p(u_{j+1}, v_*) \geq p(u_{j+1}, v_*)/2. \tag{13}$$

So as j runs from i to r, the quantity $p(u_j, v_*)$ goes from α to 1, and it at most doubles in each step. Therefore, there must be some $j > i$ such that $\alpha \cdot (r/4) \leq p(u_j, v_*) \leq \alpha \cdot (r/2)$ (recall $\alpha < 2/r$ and $r > 4$.) It follows that $u_j \in \Lambda_j$, so Λ_j is not unlikely. Let $\Lambda(v) = \Lambda_j$ and $h(v) = j - i$. □

Claim 1 (Correctness of Gen)**.** *Let v_0 be the start vertex of P. Assume $r > 4$ and $p(v_0, v_*) \geq \varepsilon$. Then there is some seed $(x, t, r_1, \ldots, r_t, y_1, \ldots, y_t)$ so that P accepts $\mathsf{Gen}(x, t, r_1, \ldots, r_t, y_1, \ldots, y_t)$.*

Proof. For a vertex v not in the last layer, define $E_v \subseteq \{0,1\}^m$ by

$$E_v = \{z : \text{starting at } v \text{ and reading } \mathsf{BaseGen}(z)|_{h(v)} \text{ reaches } \Lambda(v)\}. \tag{14}$$

Since $p(v, \Lambda(v)) \geq \frac{2}{r^2}$ and $\mathsf{BaseGen}$ has error $\frac{1}{r^2}$, $\Pr[U_m \in E_v] \geq \frac{1}{r^2}$. Therefore, by the hitting condition,

$$\Pr_x[\exists y, \mathsf{Hit}(x, y) \in E_v] \geq 1 - \frac{1}{2wr}. \tag{15}$$

There are fewer than $2wr$ vertices, so by the union bound, there is some x_* so that for every v, there is a string y_v with $\mathsf{Hit}(x_*, y_v) \in E_v$.

We now inductively define strings $y_1, y_2, \ldots$, numbers $r_1, r_2, \ldots$, and vertices $v_1, v_2, \ldots$ so that for every i, $v_i = v_*$ or $p(v_i, v_*) \geq \varepsilon \cdot (r/4)^i$. We start with the start vertex, v_0. Let $y_{i+1} = y_{v_i}$, $r_{i+1} = h(v_i)$, and $v_{i+1} =$ the vertex reached when starting at v_i and reading $\mathsf{BaseGen}(\mathsf{Hit}(x_*, y_{i+1}))|_{r_{i+1}}$. That way, $v_{i+1} \in \Lambda(v_i)$, so indeed, $v_{i+1} = v_*$ or $p(v_{i+1}, v_*) \geq p(v_i, v_*) \cdot (r/4)$. Since probabilities cannot exceed 1, the induction must terminate at $i = t$ with $v_t = v_*$ and $\varepsilon \cdot (r/4)^{t-1} \leq 1$. This implies that $t \leq \frac{\log(1/\varepsilon)}{\log(r/4)} + 1$. By construction, P accepts $\mathsf{Gen}(x_*, t, r_1, \ldots, r_t, y_1, \ldots, y_t)$. □

2.3 Seed length

In this section, we plug in explicit constructions for $\mathsf{BaseGen}$ and Hit to prove our main result:

Theorem 1 (Main result)**.** *For every w, r, ε with $r \geq \log w$, there is an ε-HSG $\mathsf{Gen} : \{0,1\}^s \to \{0,1\}^r$ for width-w, length-r ROBPs, computable in space $O(s)$, with*

$$s \leq O\left(\frac{\log(wr) \log r}{\max\{1, \log\log w - \log\log r\}} + \log(1/\varepsilon)\right). \tag{16}$$

Proof. For any m, θ, δ, Bellare, Goldreich, and Goldwasser constructed a (θ, δ)-hitter $\mathsf{Hit} : \{0,1\}^\ell \times \{0,1\}^d \to \{0,1\}^m$ with $\ell \le O(m + \log(1/\delta))$ and $d \le O(\log(1/\theta) + \log\log(1/\delta))$, easily computable in space $O(m + \log(1/\delta) + \log(1/\theta)$ [BGG93]. With the specified parameters $\theta = \frac{1}{r^2}$, $\delta = \frac{1}{2wr}$, these lengths become $\ell \le O(m + \log(wr))$ and $d \le O(\log r)$. Therefore, the seed length of our generator is bounded by

$$\underbrace{\ell}_{\text{for } x} + \underbrace{O(\log\log(1/\varepsilon))}_{\text{for } t} + \underbrace{O(\log(1/\varepsilon))}_{\text{for } r_1,\ldots,r_t} + \underbrace{O\left(\frac{d\log(1/\varepsilon)}{\log r}\right)}_{\text{for } y_1,\ldots,y_t} \le O\left(m + \log(wr/\varepsilon)\right). \tag{17}$$

Recall that m is the seed length of $\mathsf{BaseGen}$. We take $\mathsf{BaseGen}$ to be Armoni's generator [Arm98] as optimized by Kane et al. [KNW08, Theorem A.16]. Since we can tolerate error $1/r^2$ in $\mathsf{BaseGen}$, its seed length is

$$m \le O\left(\frac{\log(wr)\log r}{\max\{1, \log\log w - \log\log r\}}\right). \tag{18}$$

Armoni's generator is computable in space $O(m)$, so Gen is computable in space $O(s)$. □

3 Simulating r random bits with $r/\log^c n$ nondeterministic bits

Definition 1. For a language L, an **RL** *algorithm* with success probability $\varepsilon = \varepsilon(n)$ is a randomized log-space algorithm A that always halts such that for every $x \in \{0,1\}^*$,

$$x \in L \implies \Pr[A(x) \text{ accepts}] \ge \varepsilon \tag{19}$$

$$x \notin L \implies \Pr[A(x) \text{ accepts}] = 0. \tag{20}$$

An **RL** *algorithm* (with no success probability specified) is an **RL** algorithm with success probability $1/\operatorname{poly}(n)$. An **NL** *algorithm* is a nondeterministic log-space algorithm A that always halts such that $x \in L$ if and only if there is some sequence of nondeterministic choices causing $A(x)$ to accept.

We now show as a corollary to Theorem 1 that for log-space algorithms, r random bits can be simulated with $r/\operatorname{polylog}(n)$ nondeterministic bits.

Corollary 1. *Suppose a language L can be decided by an* **RL** *algorithm that uses at most $r = r(n)$ random bits. Then for any constant $c \in \mathbb{N}$, L can be decided by an* **NL** *algorithm that uses $O(r/\log^c n)$ nondeterministic bits.*

Proof. Let $w = \operatorname{poly}(n)$ be such that the behavior of the **RL** algorithm on an n-bit input can be modeled as a width-w, length-r ROBP P with $w \ge r$. Let C be such that the **RL** algorithm's success probability is at least $2n^{-C}$. Let $\mathsf{Gen} : \{0,1\}^{O(\log n)} \to \{0,1\}^h$ be our ε-HSG with $h = \lceil \log^{c+1} n \rceil$ and $\varepsilon = n^{-C}/w$. The **NL** algorithm repeatedly guesses[3] a seed x and feeds $\mathsf{Gen}(x)$ to an ongoing simulation of the **RL** algorithm.

The correctness of this algorithm follows inductively from the following claim. Let $L_0, L_1, \ldots, L_r$ be the layers of P, and let $v_* \in L_r$ be the accept vertex. For any vertex $v \in L_i$, there is some seed x such that if $u \in L_{i+h}$ is the vertex reached from v by reading $\mathsf{Gen}(x)$, then

$$p(u, v_*) \ge p(v, v_*) - \varepsilon. \tag{21}$$

[3] Actually, to handle the case $r < \log^{c+1} n$, the **NL** algorithm should first deterministically check whether there is some x such that after reading $\mathsf{Gen}(x)$, the **RL** algorithm has not yet halted. If not, the **NL** algorithm should halt and accept/reject depending on whether there is some x such that the **RL** algorithm accepts when reading $\mathsf{Gen}(x)$.

Proof of this claim: Let $U = \{u \in L_{i+h} : p(u, v_*) \geq p(v, v_*) - \varepsilon\}$. Then

$$p(v, v_*) = \sum_{u \in U} p(v, u) \cdot p(u, v_*) + \sum_{u \in L_{i+h} \setminus U} p(v, u) \cdot p(u, v_*) \tag{22}$$

$$\leq p(v, U) + p(v, v_*) - \varepsilon. \tag{23}$$

Therefore, $p(v, U) \geq \varepsilon$, so the correctness of Gen completes the proof. □

4 Derandomizing small-success RL algorithms

Saks and Zhou famously showed that $\mathbf{RL} \subseteq \mathbf{DSPACE}(\log^{3/2} n)$ [SZ99]. Suppose some language L merely has an **RL** algorithm with small success probability ε. By amplification, $L \in \mathbf{RSPACE}(\log(n/\varepsilon))$, so by the Saks-Zhou theorem, $L \in \mathbf{DSPACE}(\log^{3/2}(n/\varepsilon))$. In fact, Saks and Zhou showed $L \in \mathbf{DSPACE}(\log^{3/2} n + \sqrt{\log n} \log(1/\varepsilon))$ [SZ99, Theorem 3.1].

We now show as a corollary of Lemma 1 that $L \in \mathbf{DSPACE}(\log^{3/2} n + \log n \log\log(1/\varepsilon))$, an exponential improvement in terms of ε. Our derandomization smoothly interpolates between the Saks-Zhou theorem [SZ99] and Savitch's theorem $\mathbf{NL} \subseteq \mathbf{DSPACE}(\log^2 n)$ [Sav70].

Corollary 2. *Suppose a language L admits an* **RL** *algorithm with success probability $\varepsilon = \varepsilon(n)$, where $\lceil \log(1/\varepsilon) \rceil$ can be constructed in space $O(\log^{3/2} n + \log n \log\log(1/\varepsilon))$. Then*

$$L \in \mathbf{DSPACE}(\log^{3/2} n + \log n \log\log(1/\varepsilon)). \tag{24}$$

Proof. Let $w = \text{poly}(n)$ be such that the behavior of the **RL** algorithm on an n-bit input can be modeled as a width-w, length-w ROBP P with start vertex v_0 and accept vertex v_*. By the results of Saks and Zhou [SZ99], there is a deterministic algorithm A that runs in space $O(\log^{3/2} n)$ that, given vertices u, v, will distinguish between the cases $p(u, v) = 0$ and $p(u, v) \geq \frac{2}{w^3}$. Define a digraph G, where the vertices of G are the vertices of P, and we put an edge from u to v in G if $A(u, v) = 1$. To deterministically decide L, use Savitch's algorithm [Sav70] to check for the presence of a path from v_0 to v_* through G of length at most $\lceil \log(1/\varepsilon) \rceil$.

Now we prove the correctness of this algorithm. Obviously, if $p(v_0, v_*) = 0$, the algorithm will reject, so assume $p(v_0, v_*) \geq \varepsilon$. For any vertex v, $p(v, \Lambda(v)) \geq \frac{2}{w^2}$, so there is some $u \in \Lambda(v)$ so that $p(v, u) \geq \frac{2}{w^3}$. That vertex u satisfies $p(u, v_*) \geq p(v, v_*) \cdot (w/4) \geq 2p(v, v_*)$. It follows inductively that there is a path through G from v_0 to v_* of length at most $\lceil \log(1/\varepsilon) \rceil$. □

5 Directions for further research

- Is there an explicit PRG with the same seed length as our HSG? This would imply $\mathbf{BPL} \subseteq \mathbf{DSPACE}(\log^{3/2} n/\sqrt{\log\log n})$ [SZ99, Arm98, HU17], slightly improving the best known derandomization of **BPL** [SZ99].

- A less ambitious goal is to construct a *pseudorandom pseudodistribution* (PRPD). As mentioned in Section 1.3.1, Braverman et al. obtained an explicit PRPD with seed length $\widetilde{O}(\log(wr) \log r + \log(1/\varepsilon))$ [BCG18]. An explicit PRPD with the same seed length as our HSG would imply that every **BPL** algorithm using r random bits can be simulated by a **BPL** algorithm using $O(r/\log^c n)$ random bits for any constant c, improving Corollary 1.

- Braverman et al. gave a PRG for *regular* ROBPs of width $w = \text{polylog}(r)$ with seed length $\widetilde{O}(\log r \log(1/\varepsilon))$ [BRRY14]. Is there an explicit HSG for regular ROBPs of width $\text{polylog}(r)$ with seed length $\widetilde{O}(\log r + \log(1/\varepsilon))$?

- Andreev, Clementi, and Rolim showed that an explicit HSG for circuits would imply $\mathbf{P} = \mathbf{BPP}$, not just $\mathbf{P} = \mathbf{RP}$ [ACR96]. (Researchers subsequently discovered simpler proofs [BF99, ACRT99, GVW11].) Would an explicit HSG for ROBPs with seed length $O(\log(wr/\varepsilon))$ imply $\mathbf{L} = \mathbf{BPL}$? This question is perhaps related to the problem of proving $\mathbf{BPL} \subseteq \mathbf{NL}$.
- It's possible to prove a version of Lemma 1 with different parameters where $\Lambda(v)$ is a *single vertex*. It follows that the hitter in our construction can be replaced with an HSG for combinatorial rectangles [LLSZ97], provided BaseGen has error only $\frac{1}{w^2r^2}$. Does this idea have any applications?

6 Acknowledgments

The first author thanks Sumegha Garg for explaining to him the idea behind her work with Braverman and Cohen [BCG18]. We discovered our generator in the process of studying the generator by Braverman et al. [BCG18]. We thank Amnon Ta-Shma for a helpful discussion.

References

[ACR96] Alexander E. Andreev, Andrea E. F. Clementi, and José D. P. Rolim. Hitting sets derandomize BPP. In *Automata, languages and programming (Paderborn, 1996)*, volume 1099 of *Lecture Notes in Comput. Sci.*, pages 357–368. Springer, Berlin, 1996.

[ACRT99] Alexander E. Andreev, Andrea E. F. Clementi, José D. P. Rolim, and Luca Trevisan. Weak random sources, hitting sets, and BPP simulations. *SIAM Journal on Computing*, 28(6):2103–2116, 1999.

[AKS87] Miklós Ajtai, János Komlós, and Endre Szemerédi. Deterministic simulation in LOGSPACE. In *Proceedings of the 19th Annual Symposium on Theory of Computing*, pages 132–140. ACM, 1987.

[Arm98] Roy Armoni. On the derandomization of space-bounded computations. In *Proceedings of the 2nd International Workshop on Randomization and Computation*, volume 1518 of *Lecture Notes in Computer Science*, pages 47–59. Springer, Berlin, 1998.

[BCG18] Mark Braverman, Gil Cohen, and Sumegha Garg. Hitting sets with near-optimal error for read-once branching programs. In *Proceedings of the 50th Annual Symposium on Theory of Computing*, 2018. To appear.

[BDVY13] Andrej Bogdanov, Zeev Dvir, Elad Verbin, and Amir Yehudayoff. Pseudorandomness for width-2 branching programs. *Theory of Computing*, 9:283–292, 2013.

[BF99] H. Buhrman and L. Fortnow. One-sided versus two-sided error in probabilistic computation. In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1563, pages 100–109. Berlin, 1999.

[BGG93] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Randomness in interactive proofs. *Computational Complexity*, 3(4):319–354, 1993.

[BNS92] László Babai, Noam Nisan, and Márió Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *Journal of Computer and System Sciences*, 45(2):204–232, 1992.

[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.

[GMR+12] Parikshit Gopalan, Raghu Meka, Omer Reingold, Luca Trevisan, and Salil Vadhan. Better pseudorandom generators from milder pseudorandom restrictions. In *Proceedings of the 53rd Annual Symposium on Foundations of Computer Science*, pages 120–129. IEEE Computer Soc., Los Alamitos, CA, 2012.

[Gol11] Oded Goldreich. A sample of samplers: a computational perspective on sampling. In *Studies in complexity and cryptography*, volume 6650 of *Lecture Notes in Comput. Sci.*, pages 302–332. Springer, Heidelberg, 2011.

[GR14] Anat Ganor and Ran Raz. Space pseudorandom generators by communication complexity lower bounds. In *Approximation, randomization, and combinatorial optimization*, volume 28 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 692–703. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2014.

[GVW11] Oded Goldreich, Salil Vadhan, and Avi Wigderson. Simplified derandomization of BPP using a hitting set generator. In *Studies in complexity and cryptography*, volume 6650 of *Lecture Notes in Comput. Sci.*, pages 59–67. Springer, Heidelberg, 2011.

[HU17] William M. Hoza and Chris Umans. Targeted pseudorandom generators, simulation advice generators, and derandomizing logspace. In *Proceedings of the 49th Annual Symposium on Theory of Computing*, pages 629–640. ACM, New York, 2017.

[INW94] Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 356–364. ACM, 1994.

[KNW08] Daniel M Kane, Jelani Nelson, and David P Woodruff. Revisiting norm estimation in data streams. *arXiv preprint arXiv:0811.3648*, 2008.

[LLSZ97] Nathan Linial, Michael Luby, Michael Saks, and David Zuckerman. Efficient construction of a small hitting set for combinatorial rectangles in high dimension. *Combinatorica*, 17(2):215–234, 1997.

[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, 1996.

[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. $BP_HSPACE(S) \subseteq DSPACE(S^{3/2})$. *Journal of Computer and System Sciences*, 58(2):376–403, 1999.

[ŠŽ11] Jiří Šíma and Stanislav Žák. Almost k-wise independent sets establish hitting sets for width-3 1-branching programs. In *Computer science—theory and applications*, volume 6651 of *Lecture Notes in Comput. Sci.*, pages 120–133. Springer, Heidelberg, 2011.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il