SIAM J. COMPUT. © 2020 Society for Industrial and Applied Mathematics
Vol. 49, No. 4, pp. 811–820

SIMPLE OPTIMAL HITTING SETS FOR SMALL-SUCCESS RL[*]

WILLIAM M. HOZA[†] AND DAVID ZUCKERMAN[†]

Abstract. We give a simple explicit hitting set generator for read-once branching programs of width w and length r with known variable order and acceptance probability at least ε. When $r = w$, our generator has seed length $O(\log^2 r + \log(1/\varepsilon))$. When $r = \text{polylog } w$, our generator has optimal seed length $O(\log w + \log(1/\varepsilon))$. For intermediate values of r, our generator's seed length smoothly interpolates between these two extremes. Our generator's seed length improves on recent work by Braverman, Cohen, and Garg [*SIAM J. Comput.*, (2020), doi:10.1137/18M1197734]. In addition, our generator and its analysis are dramatically simpler than the work by Braverman et al. When ε is small, our generator's seed length improves on all the classic generators for space-bounded computation [N. Nisan, *Combinatorica*, 12 (1992), pp. 449–461; R. Impagliazzo, N. Nisan, and A. Wigderson, in Proceedings of the 26th Annual ACM Symposium on Theory of Computing, ACM, 1994, pp. 356–364; N. Nisan and D. Zuckerman, *J. Comput. System Sci.*, 52 (1996), pp. 43–52]. However, all of these other works construct more general objects than we do. As a corollary of our construction, 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 space bound improves on work by Saks and Zhou [*J. Comput. System Sci.*, 58 (1999), pp. 376–403], who gave an algorithm for the more general "two-sided" problem that runs in space $O(\log^{3/2} n + \sqrt{\log n}\log(1/\varepsilon))$.

Key words. pseudorandom, derandomization, hitting sets, space complexity, branching programs

AMS subject classifications. 68Q87, 68Q10, 68Q15

DOI. 10.1137/19M1268707

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 (**P** = **BPP**) and only a constant factor space blowup (**L** = **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

[*]Received by the editors June 17, 2019; accepted for publication (in revised form) June 15, 2020; published electronically July 30, 2020. A preliminary version of this paper appeared in the Proceedings of the 2018 IEEE 59th Annual Symposium on Foundations of Computer Science.
https://doi.org/10.1137/19M1268707
Funding: The first author was supported by the NSF GRFP under grant DGE-1610403 and by a Harrington Fellowship from UT Austin. The second author was supported by NSF grant CCF-1526952, NSF grant CCF-1705028, and a Simons Investigator Award (409864).
[†]Department of Computer Science, University of Texas at Austin, Austin, TX 78751 (whoza@utexas.edu, diz@cs.utexas.edu).

depending on what vertex it arrives at in the final layer. This defines a function $P : \{0,1\}^r \to \{0,1\}$.

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]

$$|\Pr[P(U_r) = 1] - \Pr[P(\mathsf{Gen}(U_s)) = 1]| \le \varepsilon. \tag{1.1}$$

A *hitting set generator* (HSG) is a relaxation of a PRG, still suitable for derandomizing **RL**, where (1.1) is replaced with

$$\Pr[P(U_r) = 1] \ge \varepsilon \implies \exists x, P(\mathsf{Gen}(x)) = 1. \tag{1.2}$$

1.2. Previous generators. We will describe only some of the myriad generators that researchers have developed for ROBPs [2, 4, 18, 15, 22, 19, 3, 16, 24, 13, 6, 11, 17, 7]. Perhaps the most famous generator is Nisan's PRG [18], which has seed length

$$O(\log(wr/\varepsilon)\log r). \tag{1.3}$$

Better generators are known when $r \ll w$, which corresponds to derandomizing algorithms that only make a few coin tosses. For $r \le O(\log^2 w/\log\log w)$ and $\varepsilon \ge 1/\operatorname{poly}(w)$, Ajtai, Komlós, and Szemerédi gave an HSG with optimal seed length $O(\log w)$ [2].[2] For $r \le \operatorname{polylog} w$ and $\varepsilon \ge 2^{-\log^{1-\Omega(1)} w}$, Nisan and Zuckerman gave a PRG with optimal seed length $O(\log w)$ [19]. Armoni [3] showed how to interpolate between Nisan's generator [18] and the Nisan–Zuckerman generator [19]. By improving an extractor in Armoni's construction, Kane, Nelson, and Woodruff [16] showed that for $r \ge \log w$, Armoni's PRG can be implemented to have seed length

$$O\left(\frac{\log(wr/\varepsilon)\log r}{\max\{1, \log\log w - \log\log(r/\varepsilon)\}}\right). \tag{1.4}$$

Uniform randomized algorithms that always halt give rise to ROBPs satisfying $w \ge r$, because the algorithm must halt within w steps to avoid repeating a configuration. However, the ROBP model is still interesting even when $w \ll r$. For the case $w = 2$, Saks and Zuckerman gave a PRG with optimal seed length $O(\log(r/\varepsilon))$ [22, 6]. For $w = 3$, Šíma and Žák gave a 0.961-HSG with seed length $O(\log r)$ [24]; Gopalan et al. gave an HSG with seed length $\widetilde{O}(\log(r/\varepsilon))$ [13]; and Meka, Reingold, and Tal recently gave a PRG with seed length $\widetilde{O}(\log r \log(1/\varepsilon))$ [17].

When $r = w$, Armoni's PRG is no better than Nisan's. Indeed, for the quarter century after Nisan announced his generator [18], there was no improvement whatsoever for the case $r = w$. In an exciting recent development, Braverman, Cohen, and Garg [7] gave an HSG (in fact, a somewhat more general object) with seed length

$$\widetilde{O}(\log(wr)\log r + \log(1/\varepsilon)). \tag{1.5}$$

Equation (1.5) improves on all prior generators when, e.g., $r = w$ and $\varepsilon = r^{-\log r}$. Unfortunately, as Braverman, Cohen, and Garg recognize in their 58-page paper, their construction is "fairly involved" and their analysis "requires a significant amount of work" [7].

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

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

1.3. Our results.

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

$$O\left(\frac{\log(wr)\log r}{\max\{1,\log\log w-\log\log r\}}+\log(1/\varepsilon)\right). \tag{1.6}$$

This seed length improves on all the classic generators [2, 4, 18, 15, 19, 3] and the recent generator by Braverman, Cohen, and Garg [7]. Our generator's seed length has optimal dependence on ε; this is the improvement over (1.4). When $r \leq \operatorname{polylog} w$, our generator has optimal seed length $O(\log w + \log(1/\varepsilon))$. This solves a problem raised by Reingold, who asked for a generator with seed length $O(\log w)$ when $r \leq \operatorname{polylog} w$ and $\varepsilon = 1/w$ [20, slide 10]. Our generator is just as interesting when r is large; for example, when $r = w$, our generator is the first with seed length $O(\log^2 r)$ for subpolynomial values of ε such as $\varepsilon = r^{-\log r}$. Our construction and analysis are very simple.

One respect in which the generator by Braverman, Cohen, and Garg [7] is superior to ours is that their construction gives a *pseudorandom pseudodistribution* (PRPD). The notion of a PRPD, introduced by Braverman, Cohen, and Garg [7], 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, Komlós, and Szemerédi [2] implies a simulation with

$$O\left(\frac{r}{\log n/\log\log n}\right) \tag{1.7}$$

nondeterministic bits. The work of Nisan and Zuckerman [19] 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. A special case of Saks and Zhou's work [21] shows 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 $\mathbf{NL} \subseteq \mathbf{DSPACE}(\log^2 n)$ [23]; indeed, our algorithm relies on Savitch's algorithm [23].

1.4. Techniques. Our results are based on an elementary structural lemma for ROBPs (Lemma 2.1). 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 $w = r$ and $\varepsilon = r^{-\log r}$. Our HSG uses a "hitter," a randomized algorithm that produces a list of $\operatorname{poly}(r)$ seeds to the given PRG. Our HSG selects $O(\log r)$ 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 [18] as the $(1/\operatorname{poly}(r))$-PRG in this construction. It has seed length $O(\log^2 r)$, so a high-quality hitter only needs $O(\log^2 r)$ random bits to produce its list. Our HSG needs an additional $O(\log^2 r)$ nondeterministic bits to select $O(\log r)$ seeds from the list. Finally, our HSG needs another $O(\log^2 r)$ 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 $O(\log^2 r)$.

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 [3, 16].

1.5. Subsequent work. After the appearance of the preliminary version of this paper, Chattopadhyay and Liao [9] gave a new construction of a PRPD (hence also an HSG) for ROBPs with seed length

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

This improves on Braverman, Cohen, and Garg's work [7] and provides another HSG with optimal dependence on ε.

Meanwhile, Ahmadinejad et al. [1] showed that it is possible to deterministically estimate the acceptance probability of a randomized log-space algorithm to within $\pm\varepsilon$ in space $O(\log^{3/2} n + \log n \log\log(1/\varepsilon))$. This generalizes our result on derandomizing small-success **RL**.

Finally, in the preliminary version of this paper, we asked whether an explicit optimal HSG for ROBPs would imply $\mathbf{L} = \mathbf{BPL}$. Cheng and the first author subsequently answered this question in the affirmative [10].

2. Our generator.

2.1. Construction. A (θ, δ)-*hitter* [12] is a function

$$\mathsf{Hit} : \{0,1\}^\ell \times \{0,1\}^d \to \{0,1\}^m \tag{2.1}$$

such that for any set $E \subseteq \{0,1\}^m$,

$$\Pr[U_m \in E] \geq \theta \implies \Pr_x[\exists y, \mathsf{Hit}(x,y) \in E] \geq 1 - \delta. \tag{2.2}$$

(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 $t \in \{1, 2, \ldots, \left\lceil \frac{\log(1/\varepsilon)}{\log(r/4)} \right\rceil + 1\}$, 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

$$\begin{aligned}(2.3)\quad \mathsf{Gen}(x, t, r_1, \ldots, r_t, y_1, \ldots, y_t) = \\ \mathsf{BaseGen}(\mathsf{Hit}(x, y_1))|_{r_1} \circ \cdots \circ \mathsf{BaseGen}(\mathsf{Hit}(x, y_t))|_{r_t}.\end{aligned}$$

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 [19].

2.2. Correctness. Let P be a width-w, length-r ROBP with layers $L_0, \ldots, L_r$. 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 $W \subseteq L_i, V \subseteq L_j$, let

$$p(W, V) = \min_{u \in W} \sum_{v \in V} p(u, v). \tag{2.4}$$

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, Komlós, and Szemerédi [2, Lemma 1].

LEMMA 2.1. *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

$$\Lambda_j = \{u \in L_j : \alpha \cdot (r/4) \leq p(u, v_*) \leq \alpha \cdot (r/2)\}. \tag{2.5}$$

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

$$\sum_{u \in \Lambda_j} p(v, u) \cdot p(u, v_*) \leq p(v, \Lambda_j) \cdot \alpha \cdot (r/2) < \frac{\alpha}{r}. \tag{2.6}$$

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,

$$p(u_j, v_*) \geq p(u_j, u_{j+1}) \cdot p(u_{j+1}, v_*) \geq p(u_{j+1}, v_*)/2. \tag{2.7}$$

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

$$\alpha \cdot (r/4) \leq p(u_j, v_*) \leq \alpha \cdot (r/2) \tag{2.8}$$

(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 2.2 (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 $(x, t, r_1, \dots, r_t, y_1, \dots, y_t)$ so that P accepts $\mathsf{Gen}(x, t, r_1, \dots, r_t, y_1, \dots, y_t)$.*

Proof. For a vertex v not in the last layer, define E_v to be the set of all $z \in \{0,1\}^m$ such that starting at v and reading $\mathsf{BaseGen}(z)|_{h(v)}$ reaches $\Lambda(v)$. 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,

$$\Pr_x[\exists y, \mathsf{Hit}(x, y) \in E_v] \geq 1 - \frac{1}{2wr}. \tag{2.9}$$

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 $y_1, y_2, \dots,$ numbers $r_1, r_2, \dots,$ and vertices $v_1, v_2, \dots$ 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 $\mathsf{Gen}(x_*, t, r_1, \dots, r_t, y_1, \dots, y_t)$. ☐

2.3. Seed length.

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

Theorem 2.3 (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*

$$s \leq O\left(\frac{\log(wr)\log r}{\max\{1, \log\log w - \log\log r\}} + \log(1/\varepsilon)\right). \tag{2.10}$$

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 $\ell \leq O(m + \log(1/\delta))$ and $d \leq O(\log(1/\theta) + \log\log(1/\delta))$, easily computable in space $O(m + \log(1/\delta) + \log(1/\theta))$ [5]. With the specified parameters $\theta = \frac{1}{r^2}$, $\delta = \frac{1}{2wr}$, these lengths become $\ell \leq O(m + \log(wr))$ and $d \leq O(\log r)$. Therefore, the seed length of our generator is bounded by

$$s \leq \underbrace{\ell}_{\text{for } x} + \underbrace{O(\log\log(1/\varepsilon))}_{\text{for } t} + \underbrace{O(\log(1/\varepsilon))}_{\text{for } r_1,\dots,r_t} + \underbrace{O\left(\frac{d\log(1/\varepsilon)}{\log r}\right)}_{\text{for } y_1,\dots,y_t} \tag{2.11}$$

$$\leq O\left(m + \log(wr/\varepsilon)\right). \tag{2.12}$$

Recall that m is the seed length of $\mathsf{BaseGen}$. We take $\mathsf{BaseGen}$ to be Armoni's generator [3] as optimized by Kane, Nelson, and Woodruff [16, Theorem A.16]. Since we can tolerate error $1/r^2$ in $\mathsf{BaseGen}$, its seed length is

$$m \leq O\left(\frac{\log(wr)\log r}{\max\{1, \log\log w - \log\log r\}}\right). \tag{2.13}$$

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.

In this section and the next, n denotes the length of the input to the space-bounded algorithm under discussion.

DEFINITION 3.1. *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 $x \in \{0,1\}^*$,*

$$x \in L \implies \Pr[A(x) \text{ accepts}] \geq \varepsilon, \tag{3.1}$$

$$x \notin L \implies \Pr[A(x) \text{ accepts}] = 0. \tag{3.2}$$

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 2.3 that for log-space algorithms, r random bits can be simulated with $r/\operatorname{polylog}(n)$ nondeterministic bits.

COROLLARY 3.2. *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 $w \geq r$. 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

$$p(u, v_*) \geq p(v, v_*) - \varepsilon. \tag{3.3}$$

Proof of this claim: Define

$$W = \{u \in L_{i+h} : p(u, v_*) \geq p(v, v_*) - \varepsilon\}. \tag{3.4}$$

Then

$$p(v, v_*) = \sum_{u \in W} p(v, u) \cdot p(u, v_*) + \sum_{u \in L_{i+h} \setminus W} p(v, u) \cdot p(u, v_*) \tag{3.5}$$

$$\leq p(v, W) + p(v, v_*) - \varepsilon. \tag{3.6}$$

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

4. Derandomizing small-success RL algorithms.

Saks and Zhou famously showed that $\mathbf{BPL} \subseteq \mathbf{DSPACE}(\log^{3/2} n)$ [21]. Suppose some language L merely has an **RL** algorithm with small success probability ε. By amplification,

$$L \in \mathbf{RSPACE}(\log(n/\varepsilon)), \tag{4.1}$$

[3]Actually, to handle the case $r(n) < \log^{c+1} n$, the **NL** algorithm should first deterministically check whether it is the case that for every x, after reading $\mathsf{Gen}(x)$, the **RL** algorithm has halted. If so, the **NL** algorithm should halt and accept/reject depending on whether there is some x such that the **RL** algorithm accepts when reading $\mathsf{Gen}(x)$.

so by the Saks–Zhou theorem, $L \in \mathbf{DSPACE}(\log^{3/2}(n/\varepsilon))$. In fact, Saks and Zhou showed $L \in \mathbf{DSPACE}(\log^{3/2} n + \sqrt{\log n}\log(1/\varepsilon))$ [21, Theorem 3.1].

We now show as a corollary of Lemma 2.1 that

$$L \in \mathbf{DSPACE}(\log^{3/2} n + \log n \log\log(1/\varepsilon)), \tag{4.2}$$

an exponential improvement in terms of ε. Our derandomization smoothly interpolates between the Saks–Zhou theorem [21] and Savitch's theorem [23]

$$\mathbf{NL} \subseteq \mathbf{DSPACE}(\log^2 n). \tag{4.3}$$

Corollary 4.1. *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*

$$O(\log^{3/2} n + \log n \log\log(1/\varepsilon)). \tag{4.4}$$

Then

$$L \in \mathbf{DSPACE}(\log^{3/2} n + \log n \log\log(1/\varepsilon)). \tag{4.5}$$

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 [21], 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 [23] 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 $\mathbf{BPL} \subseteq \mathbf{DSPACE}(\log^{3/2} n/\sqrt{\log\log n})$ [21, 3, 14], slightly improving the best known derandomization of **BPL** [21].
- A less ambitious goal is to construct a PRPD. As mentioned in section 1.3.1, Braverman, Cohen, and Garg obtained an explicit PRPD with seed length $\widetilde{O}(\log(wr)\log r + \log(1/\varepsilon))$ [7]. 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 3.2.
- Braverman et al. gave a PRG for *regular* ROBPs of width $w = \text{polylog}(r)$ with seed length $\widetilde{O}(\log r \log(1/\varepsilon))$ [8]. Is there an explicit HSG for regular ROBPs of width polylog(r) with seed length $\widetilde{O}(\log r + \log(1/\varepsilon))$?

Acknowledgments. The first author thanks Sumegha Garg for explaining to him the idea behind her work with Braverman and Cohen [7]. We discovered our generator in the process of studying the generator by Braverman, Cohen, and Garg [7]. We thank Amnon Ta-Shma for a helpful discussion.

REFERENCES

[1] A. Ahmadinejad, J. Kelner, J. Murtagh, J. Peebles, A. Sidford, and S. Vadhan, *High-Precision Estimation of Random Walks in Small Space*, preprint, arXiv:1912.04524[cs.CC], 2019, https://arxiv.org/abs/1912.04524.

[2] M. Ajtai, J. Komlós, and E. Szemerédi, *Deterministic simulation in LOGSPACE*, in Proceedings of the 19th Annual Symposium on Theory of Computing, ACM, 1987, pp. 132–140.

[3] R. Armoni, *On the derandomization of space-bounded computations*, in Proceedings of the 2nd International Workshop on Randomization and Computation, Lecture Notes in Comput. Sci. 1518, Springer, Berlin, 1998, pp. 47–59, https://doi.org/10.1007/3-540-49543-6_5.

[4] L. Babai, N. Nisan, and M. Szegedy, *Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs*, J. Comput. System Sci., 45 (1992), pp. 204–232, https://doi.org/10.1016/0022-0000(92)90047-M.

[5] M. Bellare, O. Goldreich, and S. Goldwasser, *Randomness in interactive proofs*, Comput. Complexity, 3 (1993), pp. 319–354, https://doi.org/10.1007/BF01275487.

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

[7] M. Braverman, G. Cohen, and S. Garg, *Pseudorandom pseudo-distributions with near-optimal error for read-once branching programs*, SIAM J. Comput., (2020), https://doi.org/10.1137/18M1197734.

[8] M. Braverman, A. Rao, R. Raz, and A. Yehudayoff, *Pseudorandom generators for regular branching programs*, SIAM J. Comput., 43 (2014), pp. 973–986, https://doi.org/10.1137/120875673.

[9] E. Chattopadhyay and J.-J. Liao, *Optimal error pseudodistributions for read-once branching programs*, in 35th Computational Complexity Conference (CCC 2020), LIPIcs. Leibniz Int. Proc. Inform. 169, S. Saraf, ed., Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2020, pp. 25:1–25:27, https://doi.org/10.4230/LIPIcs.CCC.2020.25.

[10] K. Cheng and W. M. Hoza, *Hitting sets give two-sided derandomization of small space*, in 35th Computational Complexity Conference (CCC 2020), LIPIcs. Leibniz Int. Proc. Inform. 169, S. Saraf, ed., Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2020, pp. 10:1–10:25, https://doi.org/10.4230/LIPIcs.CCC.2020.10.

[11] A. Ganor and R. Raz, *Space pseudorandom generators by communication complexity lower bounds*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014), K. Jansen, J. D. P. Rolim, N. R. Devanur, and C. Moore, eds., LIPIcs. Leibniz Int. Proc. Inform. 28, Dagstuhl, Germany, 2014, pp. 692–703, https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2014.692.

[12] O. Goldreich, *A sample of samplers: A computational perspective on sampling*, in Studies in Complexity and Cryptography, O. Goldreich, ed., Lecture Notes in Comput. Sci. 6650, Springer, Heidelberg, 2011, pp. 302–332, https://doi.org/10.1007/978-3-642-22670-0_24.

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

[14] W. M. Hoza and C. Umans, *Targeted pseudorandom generators, simulation advice generators, and derandomizing logspace*, in Proceedings of the 49th Annual Symposium on Theory of Computing, ACM, New York, 2017, pp. 629–640.

[15] R. Impagliazzo, N. Nisan, and A. Wigderson, *Pseudorandomness for network algorithms*, in Proceedings of the 26th Annual ACM Symposium on Theory of Computing, ACM, 1994, pp. 356–364.

[16] D. M. Kane, J. Nelson, and D. P. Woodruff, *Revisiting Norm Estimation in Data Streams*, preprint, arXiv:0811.3648[cs.DS], 2008, https://arxiv.org/abs/0811.3648.

[17] R. Meka, O. Reingold, and A. Tal, *Pseudorandom generators for width-*3 *branching programs*, in Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, ACM, New York, 2019, pp. 626–637, https://doi.org/10.1145/3313276.3316319.

[18] N. Nisan, *Pseudorandom generators for space-bounded computation*, Combinatorica, 12 (1992), pp. 449–461, https://doi.org/10.1007/BF01305237.

[19] N. Nisan and D. Zuckerman, *Randomness is linear in space*, J. Comput. System Sci., 52 (1996), pp. 43–52, https://doi.org/10.1006/jcss.1996.0004.

[20] O. Reingold, *Randomness vs. Memory: Prospects and Barriers*, Presented at "Barriers in Computational Complexity" workshop, 2010, https://omereingold.files.wordpress.com/2014/10/rlbarriers.pptx.

[21] M. Saks and S. Zhou, $\mathrm{BP_H SPACE}(S) \subseteq \mathrm{DSPACE}(S^{3/2})$, J. Comput. System Sci., 58 (1999), pp. 376–403, https://doi.org/10.1006/jcss.1998.1616.

[22] M. Saks and D. Zuckerman, unpublished, 1995.

[23] W. J. Savitch, *Relationships between nondeterministic and deterministic tape complexities*, J. Comput. System. Sci., 4 (1970), pp. 177–192, https://doi.org/10.1016/S0022-0000(70)80006-X.

[24] J. Šíma and S. Žák, *Almost k-wise independent sets establish hitting sets for width*-3 1-*branching programs*, in Computer science—Theory and Applications, Lecture Notes in Comput. Sci. 6651, Springer, Heidelberg, 2011, pp. 120–133, https://doi.org/10.1007/978-3-642-20712-9_10.