On Probabilistic Space-Bounded Machines with Multiple Access to Random Tape

Debasis Mandal[1], A. Pavan[1], and N. V. Vinodchandran[2]

[1] Department of Computer Science, Iowa State University
{debasis,pavan}@cs.iastate.edu*
[2] Department of Computer Science and Engineering, University of Nebraska-Lincoln
vinod@cse.unl.edu**

Abstract. We investigate probabilistic space-bounded Turing machines that are allowed to make multiple passes over the random tape. As our main contribution, we establish a connection between derandomization of such probabilistic space-bounded classes to the derandomization of probabilistic time-bounded classes. Our main result is the following.

- For some integer $k > 0$, if all the languages accepted by bounded-error randomized log-space machines that use $O(\log n \log^{(k+3)} n)$ random bits and make $O(\log^{(k)} n)$ passes over the random tape is in deterministic polynomial-time, then $\mathrm{BPTIME}(n) \subseteq \mathrm{DTIME}(2^{o(n)})$. Here $\log^{(k)} n$ denotes log function applied k times iteratively.

This result can be interpreted as follows: If we restrict the number of random bits to $O(\log n)$ for the above randomized machines, then the corresponding set of languages is trivially known to be in P. Further, it can be shown that (proof is given in the main body of the paper) if we instead restrict the number of passes to only $O(1)$ for the above randomized machines, then the set of languages accepted is in P. Thus our result implies that any non-trivial extension of these simulations will lead to a non-trivial and unknown derandomization of $\mathrm{BPTIME}(n)$. Motivated by this result, we further investigate the power of multi-pass, probabilistic space-bounded machines and establish additional results.

1 Introduction

In this paper we investigate probabilistic space-bounded Turing machines that are allowed to access their random bits multiple times. In the traditional definition of probabilistic space-bounded computations, a probabilistic machine can access its random tape in a *one-way*, read-only manner and the random tape does not count towards the space complexity of the probabilistic machine. In particular, the machine cannot reread the random bits unless they are stored in its work tapes. This access mechanism is the most natural one as it corresponds to modeling probabilistic machines as coin-tossing machines, originally defined by Gill [8]. The complexity class BPL is the class of languages accepted by

* Research Supported in part by NSF grants 0916797, 1421163.
** Research Supported in part by NSF grants 0916525, 1422668.

such bounded-error probabilistic machines that use logarithmic space and halt in polynomial time[3]. The class RL is its one-sided error counterpart. Whether BPL or even RL can be derandomized to deterministic log-space is one of the central questions in the study of space-bounded computation. In spite of clear and steady progress in space-bounded derandomization, this question is far from settled [4, 11, 15, 17, 18, 19, 21].

Even though one-way access to the random tape is the standard in investigating probabilistic space-bounded computations, researchers have considered space-bounded models where the base probabilistic machines are allowed to read contents of the random tape multiple times [3, 5, 6, 16]. However, our understanding of such multiple-access models is limited. For example, consider the class of languages that are accepted by bounded-error probabilistic log-space machines that have an unrestricted *two-way* access to the random tape (we denote this class by 2-*way*BPL). While we know that BPL is in deterministic polynomial time, it is not known whether 2-*way*BPL is even in deterministic sub-exponential time (note that it is in BPP). This is because, while one-way access machines can be characterized using certain graph reachability problem, we do not have such a nice combinatorial characterization for two-way access machines. It is also interesting to note that allowing two way access to the random tape for a space-bounded machine makes the corresponding nonuniform classes more closer to randomized circuit complexity classes. It is known that log-space uniform NC_1 is in deterministic log-space, where NC_1 is the class of languages accepted by polynomial-size, bounded fan-in $O(\log n)$-depth circuits. However, a randomized version of this inclusion is not known to hold. That is, we do not know whether (uniform) BPNC_1 is contained in BPL. However, it is a folklore that (uniform) BPNC_1 is in 2-*way*BPL (for example, see [16]).

This paper revisits probabilistic space-bounded machines that are allowed to access their random bits multiple times. In particular, we study a model where the probabilistic space-bounded machines are allowed to make *multiple passes* over the random tape, where in each pass the machines access their random tapes in a traditional one-way manner. This model was first considered by David, Papakonstantinou, and Sidiropoulos [6]. Clearly, the *multi-pass* model is intermediate between the standard model and the two-way access model. Our focus is to investigate the consequences of *derandomizing* such machines. Our main conceptual contribution is that derandomizing such probabilistic space-bounded machines leads to derandomization of probabilistic *time-bounded* classes.

Our Results

As our main result, we show a connection between derandomization of multi-pass probabilistic space-bounded machines and derandomization of probabilistic (linear) time-bounded machines. In particular, we prove the following theorem.

[3] We only consider probabilistic machines that halt on all inputs on all settings of the random tape. If we do not require the machines to halt, then we get a potentially larger class of languages [13, 20].

Theorem 1. *For some constant $k > 0$, if every language decided by a bounded-error probabilistic log-space machine that uses $O(\log n \log^{(k+3)} n)$ random bits and makes $O(\log^{(k)} n)$ passes over its random tape is in* P*, then* $\mathrm{BPTIME}(n) \subseteq \mathrm{DTIME}(2^{o(n)})$.

Here $\log^{(k)} n$ denotes log function applied k times iteratively. Showing that $\mathrm{BPTIME}(n)$ is a subset of $\mathrm{DTIME}(2^{o(n)})$ is a significant open problem. The best unconditional derandomization of $\mathrm{BPTIME}(n)$ till date is due to Santhanam and van Melkebeek [22] who showed that any bounded-error linear time probabilistic machine can be simulated in $\mathrm{DTIME}(2^{\epsilon n})$, where $\epsilon > 0$ is a constant that depends on the number of tapes and the alphabet size of the probabilistic machine. Here we show that derandomizing a slightly non-constant pass probabilistic space-bounded machine that uses slightly larger than $O(\log n)$ random bits yields a non-trivial derandomization of $\mathrm{BPTIME}(n)$. Notice that if we restrict the number of random bits from $O(\log n \log^{(k+3)}(n))$ to $O(\log n)$, then the corresponding set of languages is trivially in P. If we restrict the number of passes from $O(\log^{(k)} n)$ to $O(1)$, we can still show that the set of languages accepted is in P (refer to Section 3.2 for a proof). Thus, the above theorem states that any extension of these simulations will lead to a non-trivial and unknown derandomization of $\mathrm{BPTIME}(n)$.

We also present some upper bounds on the class of languages accepted by multi-pass probabilistic space-bounded machines. Even though we are unable to prove that the hypothesis of Theorem 1 holds, we show that for every constants $k \geq 3$ and $\epsilon > 0$, languages accepted by probabilistic log-space machines that use $O(\log n \log^{(k+3)} n)$ random bits and make $O(\log^{(k)} n)$ passes over its random tapes are in $\mathrm{DSPACE}(\log n (\log \log n)^{1/2+\epsilon})$, which in turn is contained in $\mathrm{DTIME}(n^{(\log \log n)^{1/2+\epsilon}})$. We also show that any $k(n)$-pass, $s(n)$-space, probabilistic machine can be simulated by traditional $k(n)s(n)$-space bounded probabilistic machines. Thus, in particular, a constant number of passes do not add power to the traditional one-way random tape machines.

Finally, we extend some well-known results regarding standard probabilistic log-space classes to multi-pass, probabilistic log-space classes.

Prior Work on Multiple Access Models

As mentioned earlier, the literature on probabilistic space-bounded Turing machines with multiple access to random bits is limited compared to the standard model. Borodin, Cook, and Pippenger [3], while investigating deterministic simulations of probabilistic space-bounded classes, raised the question whether two-way probabilistic $s(n)$-space-bounded machines can be simulated deterministically in $O(s^2(n))$ space. Karpinsky and Verbeek [13] showed that the answer is negative in general. They showed that two-way log-space probabilistic machines that are allowed to run for time $2^{n^{O(1)}}$ time can simulate PSPACE with zero error probability. Another relevant result is due to Nisan [16] who showed BPL can be simulated by zero-error, probabilistic, space-bounded machines that has a two-way access to the random tape. Nisan also showed that 2-*way*BPL is same as almost-logspace, where almost-logspace is the class of languages accepted by deterministic log-space machines relative to a random oracle.

Probabilistic space-bounded machines that can make multiple passes over the random tape was first considered by David, Papakonstantinou, and Sidiropoulos [6]. They showed that any pseudo-random generator that fools traditional $k(n)s(n)$-space bounded machines can also fool $k(n)$-pass $s(n)$-space bounded machines. As a corollary, they obtain that polylog-pass, randomized log-space is contained in deterministic polylog-space. David, Nguyen, Papakonstantinou, and Sidiropoulos [5] considered probabilistic space-bounded machine that have access to a stack and a two-way/multiple pass access to the random tape and related them to traditional classes.

2 Preliminaries

We assume familiarity with the standard complexity classes [2]. We are interested in probabilistic space-bounded machines that can access random bits multiple times. For such machines, the random bits appear on a special read-only tape called the *random tape*. In addition to the random tape, these machines have one read-only input tape and few read-write work tapes, as standard in space-bounded machine models. The total space used by the work tapes signify the space bound of such a machine. We also assume that all the probabilistic space-bounded machines halt in time at most exponential in their space bounds. Thus, the number of random bits used by them is at most exponential in their space-bounds. More formally, our multi-pass machines are defined as follows:

Definition 1. *A language L is in $k(n)$-pass* BPSPACE$[s(n), r(n)]$ *if there exists an $O(s(n))$-space bounded probabilistic Turing machine M such that on any input x of length n:*

- *M makes $k(n)$ passes over the random tape, during each pass it accesses the contents of random tape in a* one-way *manner,*
- *M uses at most $O(r(n))$ random bits, and*
- *the probability that M incorrectly decides x is at most* $1/3$.

In our notation, BPL $=$ 1-*pass* BPSPACE$[\log n, n^{O(1)}]$. In Section 3, we observe that a constant factor in the number of passes does not add power to the model and hence $O(1)$-*pass* BPSPACE$[\log n, n^{O(1)}]$ is also same as BPL.

Since we assume that every space-bounded probabilistic machine halts on all settings of random tape on all inputs, the running time of the multi-pass machine is bounded by $2^{O(s(n))}$ where $s(n)$ is the space bound. Thus, this machine can only access random tape of length $2^{O(s(n))}$. Indeed, when the number of random bits is exponential in space, *i.e.*, $r(n) = 2^{O(s(n))}$, we simply write the above class as $k(n)$-*pass* BPSPACE$(s(n))$. Further, when the the space of the $k(n)$-pass BPSPACE machine is bounded by $O(\log n)$, we simply write the class as $k(n)$-*pass* BPL$[r(n)]$.

$\log^{(k)}(\cdot)$ is the iterated logarithmic function applied k times with itself.

3 Space-bounded Machines with Multiple Passes over the Random Tape

In this section, we consider probabilistic space-bounded machines that are allowed to make multiple passes over the random tape. First, we establish results that connect derandomization of such machines to derandomization of probabilistic time classes. Next, we consider the problem of simulating multi-pass, probabilistic, space-bounded machines with the traditional one-pass machines. Finally, we provide a space-efficient deterministic nonuniform simulation of the multi-pass machines.

3.1 Derandomization of Probabilistic Time

As our main result of this section, we show that a time-efficient derandomization of probabilistic log-space machines that use very few random bits and make very few passes over their random tape, yields a non-trivial derandomization of probabilistic time. In particular, we show the following theorem.

Theorem 2. *If for some constant* $k > 0$*,* $O(\log^{(k)} n)$*-pass* $\mathrm{BPL}\left[\log n \log^{(k+3)} n\right]$ *is in* P*, then* $\mathrm{BPTIME}(n) \subseteq \mathrm{DTIME}(2^{o(n)})$.

Remark. We use the iterated logarithmic function for simplicity, but the above theorem can be proved with any "nice" slowly growing function $f(n) \in \omega(1)$.

We establish the theorem by first proving that every $\mathrm{BPTIME}(n)$ machine can be simulated by a bounded-error probabilistic space-bounded machine that makes $O(\log^{(k)} n)$ passes over the random tape and uses $o(n)$ space, on inputs of size n. There is a trade-off between the number of passes and space used by the simulating machine and this trade-off is essential in the proof. More formally, we prove the following theorem.

Theorem 3. *For every constant* $k > 0$*,*

$$\mathrm{BPTIME}(n) \subseteq O(\log^{(k)} n)\text{-}\mathit{pass}\ \mathrm{BPSPACE}\left[n/\log^{(k+3)} n, n\right].$$

Remark. The above theorem may be of independent interest. Hopcroft, Paul, and Valiant [10] showed that $\mathrm{DTIME}(n) \subseteq \mathrm{DSPACE}(o(n))$. The analogous inclusion relationship for probabilistic classes is not known. That is, we do not know unconditionally if $\mathrm{BPTIME}(n)$ is a subset of $\mathrm{BPSPACE}(o(n))$ (see [12, 14] for conditional results in this direction). The above theorem can be viewed as a partial solution; if we allow the space-bounded machine to have a slightly non-constant number of passes, then $\mathrm{BPTIME}(n)$ can be simulated in $o(n)$ probabilistic space.

We first give the proof of Theorem 2 assuming Theorem 3. The proof uses a simple padding argument.

Proof of Theorem 2. Let $k > 0$ be a constant for which the hypothesis in the statement of Theorem 2 holds. Let L be a language in BPTIME(n). By Theorem 3, L is in $O(\log^{(k-1)} n)$-*pass* BPSPACE $\left[n/\log^{(k+2)} n, n\right]$. Let

$$L' = \left\{ \langle x, 0^{2^{n/\log^{(k+2)} n} - n} \rangle \;\mid\; x \in L, |x| = n \right\}.$$

It is easy to see that L' is in $O(\log^{(k)} n)$-*pass* BPL $\left[\log n \log^{(k+3)} n\right]$. By our hypothesis, L' is in P. So $L' \in \mathrm{DTIME}(n^{\ell})$ for some $\ell > 0$. From this it follows that for some $\ell > 0$, L is in DTIME $\left(2^{\frac{\ell n}{\log^{(k+2)} n}}\right)$ and thus in DTIME$(2^{o(n)})$. $\square$

Now we move on to proving Theorem 3. The proof relies on the classical result of Hopcroft, Paul, and Valiant [10] who showed that every deterministic machine running in time $O(n)$ can be simulated by a deterministic machine that uses $O(n/\log n)$ space. If we adopt their proof to the case of probabilistic machines, we obtain that every bounded-error probabilistic machine running in time $O(n)$ can be simulated by a bounded-error, probabilistic machine that uses space $O(n/\log n)$; however the simulating machine makes an exponential number of passes over the random tape. We observe that the number of passes can be greatly reduced at the expense of a little increase in space. This is essentially achieved by using a careful choice of parameters than those used in [10].

To proceed with the proof we need the notions of *block-respecting Turing machines* and *pebbling games* [10].

Definition 2. *Let M be a multi-tape Turing machine running in time $t(n)$. Let $b(n)$ be a function such that $1 \le b(n) \le t(n)/2$. Divide the computation of M into $a(n)$ time segments so that each segment has $b(n) = t(n)/a(n)$ steps. Also divide each tape of M into $a(n)$ blocks so that each block has exactly $b(n)$ cells. We say that the machine M is $b(n)$-*block respecting *if during each time segment every tape head of the machine M visits cells of exactly one block. I.e, a tape head can cross a block boundary only at time step $c \cdot b(n)$ for some integer $c > 0$.*

Hopcroft, Paul, and Valiant showed that every ℓ-tape Turing machine running in time $t(n)$ can be simulated by a $(\ell + 1)$ tape $b(n)$-block respecting Turing machine running in time $O(t(n))$ for any $b(n)$ such that $1 \le b(n) \le t(n)/2$.

Pebbling Game. Let G be a directed acyclic graph and w be a special vertex of the graph. We say that a vertex u is a predecessor of vertex v if there is an edge from u to v. The goal is to place a pebble on the special vertex w using as little pebbles as possible, subject to the following constraints: we can place a pebble on a vertex v only if all predecessors of v have pebbles on them, a pebble can be removed from a vertex at any time.

Hopcroft, Paul, and Valiant showed (by means of a clever divide-and-conquer algorithm) that every bounded-degree graph with n vertices can be pebbled using $O(n/\log n)$ pebbles, and there is a deterministic algorithm S that does this pebbling in time $O(2^{n^2})$. Now we are ready to prove Theorem 3.

Proof of Theorem 3. Fix $k > 0$, and set $b(n) = n/\log^{(k+2)} n$. Let L be a language in BPTIME(n) and let M be an ℓ-tape, $b(n)$-block respecting probabilistic machine that accepts L in time $t(n) = O(n)$. Set $a(n) = t(n)/b(n)$. Without loss of generality, we can assume that M reads the contents of the random-tape in a one-way manner. The computation graph G_M of M is an edge-labeled graph defined as follows. The vertex set is $V = \{1, \cdots, a(n)\}$. For $1 \leq i < a(n)$, $\langle i, i+1 \rangle \in E$ with label 0, implying the computation at time segment $i+1$ requires the computation of the time segment i. Assume that the tape heads are numbered $1, \cdots, \ell$. We place an edge from i to j with label $h \in [1, \ell]$ if the following holds: Suppose that during the time segment i the tape head h is in some block b and the next time segment that the tape head h revisits block b is j (*i.e.*, the computation at time segment j requires the content of the block b from the time-segment i). This process defines a multi-graph.

Given $1 \leq i \leq a(n)$, let $B_1(i), \cdots, B_\ell(i)$ be the blocks that each of the ℓ tape heads are visiting during time segment i. Let $C(i)$ be a string that describes the contents of blocks $B_1(i), \cdots, B_\ell(i)$ and the state q at the end of time segment i. The following observation is crucial.

Observation 1. *Suppose a vertex j has predecessors $i_1, \cdots, i_r$ ($1 \leq r \leq \ell$). Then we can simulate the computation of M during time segment j by knowing $C(i_1), \cdots, C(i_r)$. Thus we can compute $C(j)$.*

Using this observation, as in [10], we simulate M by a machine M' as follows. We describe a simulation algorithm that gets a bounded degree graph (degree $\leq \ell + 1$) G as input and attempts to simulate M.

> Call the pebbling algorithm S on graph G. If S places pebble on vertex i, then compute $C(i)$ and store $C(i)$. If S removes pebble from a vertex i, then erase $C(i)$.

Note that a priori, we do not know the correct computation graph G_M. We will first assume that the correct computation graph G_M is known to us and thus can be given as input to the above algorithm. Latter we will remove this restriction. By Observation 1, it follows that the above algorithm correctly simulates M (under the assumption that G_M is known). Now we bound the space, time, number of passes, and number of random bits required by the above simulation algorithm (under the assumption that G_M is known). We start with the following two claims whose proofs appear in the appendix.

Claim 1. The total space used by the above simulation is $O\left(\frac{a(n)b(n)}{\log a(n)} + 2^{a^2(n)}\right)$.

Claim 2. The above simulation algorithm makes at most $O(2^{a^2(n)})$ passes over the random tape.

Now we address the assumption that the computation graph G_M is known. The proof of following observation appears in the appendix.

Observation 2. *Suppose that G is not the correct computation graph. If the above simulation algorithm gets G as input, then it will discover that G is not the correct computation graph, using $O(\frac{a(n)b(n)}{\log a(n)} + 2^{a^2(n)})$ space and by making $O(2^{a^2(n)})$ passes over the random tape.*

So our final simulation of M proceeds as follows: Iterate through all possible computation graphs; for each graph G, attempt to simulate M using the above algorithm. If it discovers that G is not a correct computation graph, then proceed to next graph.

By Claim 1 and Observation 2, each iteration needs $O(\frac{a(n)b(n)}{\log b(n)} + 2^{a^2(n)})$ space. Since we can reuse space from one iteration to the next iteration, total space is bounded by $O(\frac{a(n)b(n)}{\log b(n)} + 2^{a^2(n)})$. By Claim 2 and Observation 2, each iteration can be done making $2^{a^2(n)}$ passes. Since there are at most $2^{a^2(n)}$ possible computation graphs, the total number of passes is $2^{2a^2(n)}$.

By plugging in the values of $a(n)$ and $b(n)$ from above, we obtain that the space used by the simulating machine is $O(n/\log^{(k+3)} n)$ and the number of passes is $O(\log^{(k)} n)$. Finally, note that the number of random bits used by the simulating machine remains same as the number of random bits used by M, i.e., $O(n)$. This completes the proof of the Theorem. □

3.2 Simulating Multiple Passes with Single Pass

An obvious question at this point is the following: Can we simulate multi-pass probabilistic machines with traditional one-pass probabilistic space bounded machines? The main result of this subsection shows that passes can be traded for space. This helps us to obtain an upper bound on the deterministic space to simulate a multi-pass probabilistic space-bounded machine. We first start with the following lemma whose proof appears in the appendix.

Lemma 1. *If a language L is in $k(n)$-pass* BPSPACE$[s(n), r(n)]$*, then there is a probabilistic $O(k(n)s(n))$-space bounded machine N that has one-way access to the random tape and for every $x \in \Sigma^n$,*

$$\Pr[N(x) = L(x)] \geq \frac{1}{2} + \frac{1}{2^{O(k(n)s(n))}}.$$

Moreover, N uses $O(r(n) + k(n)s(n))$ random bits.

Using above Lemma, we obtain the following.

Theorem 4. *$k(n)$-pass* BPSPACE$(s(n)) \subseteq$ BPSPACE$(k(n)s(n))$.

Proof. Let L be a language that is accepted by a $k(n)$-pass BPSPACE$(s(n))$ machine M. By definition, this machine uses $2^{O(s(n))}$ random bits. Thus by Lemma 1, there is an $O(k(n)s(n))$-space bounded, one-pass, probabilistic machine N that uses $O(2^{O(s(n))} + k(n)s(n))$ random bits, and

$$Pr[L(x) = N(x)] \geq \frac{1}{2} + \frac{1}{2^{O(k(n)s(n))}}.$$

We can amplify the success probability of N to $2/3$ by simulating it $2^{O(k(n)s(n))}$ times and taking the majority vote. This will use $2^{O(k(n)s(n))}$ random bits. Thus we obtain a $O(k(n)s(n))$-space bounded machine that uses $2^{O(k(n)s(n))}$ random bits. Thus L is in BPSPACE$(k(n)s(n))$. $\square$

Corollary 1. $O(1)$-*pass* BPL = BPL.

We relate the above lemma to the result of [6].

Theorem 5. *Let M be a $k(n)$-pass, $s(n)$-space bounded machine that uses $r(n)$ random bits. Any pseudo-random generator that fools $k(n)s(n)$-space bounded machines (that read their input in a one-way manner) running on $r(n)$-bit input strings also fools M.*

Their result states that to deterministically simulate $k(n)$-pass, $s(n)$-space bounded probabilistic machines, a pseudo-random generator against standard $O(k(n)s(n))$-space bounded probabilistic machine suffices. Lemma 1 can be interpreted as an explanation of their result, as it shows that any $k(n)$-pass, $s(n)$-space bounded machine can indeed be simulated by a standard $O(k(n)s(n))$ space bounded machine.

Next, we consider the main result of this subsection: deterministic simulation of the class $k(n)$-pass BPSPACE$(s(n))$. By above theorem, this is a subclass of BPSPACE$(k(n)s(n))$. By the celebrated results of Nisan [15] and Saks and Zhou [21], it follows that this class is a subset of DSPACE$(k^{3/2}(n)s^{3/2}(n))$. Observe below that we can get rid of the polynomial factor off the number of passes, more formally,

Theorem 6. $k(n)$-*pass* BPSPACE$(s(n)) \subseteq$ DSPACE$(k(n)s^{3/2}(n))$.

Proof. Let L be a language that is accepted by a $k(n)$-pass BPSPACE$(s(n))$ machine M. Since M halts in $2^{O(s(n))}$ time, $k(n)$ is bounded by $2^{O(s(n))}$ and M uses $2^{O(s(n))}$ random bits. Thus, by Lemma 1, there is an $O(k(n)s(n))$-space bounded, one-pass, probabilistic machine N that uses $2^{O(s(n))}$ random bits and

$$Pr[L(x) = N(x)] \geq \frac{1}{2} + \frac{1}{2^{O(k(n)s(n))}}.$$

Saks and Zhou [21], building on Nisan's [15] work showed that any language accepted by a probabilistic machine using $O(s(n))$-space, $2^{O(r(n))}$ random bits, with success probability as low as $1/2 + 1/2^{O(s(n))}$, is in deterministic space $O(s(n)r^{1/2}(n))$. Applying this to our case, we obtain that N can be simulated by a deterministic space-bounded machine that uses $O(k(n)s^{3/2}(n))$. $\square$

Recall that our hypothesis in Theorem 2 states that for some $k > 0$ if the complexity class $\log^{(k)} n$-*pass* BPL$(\log n \log^{(k+2)} n)$ is in P. We obtain the following upper bound for this class, by applying Lemma 1 and the techniques of Saks and Zhou [21]. The proof is similar to the proof of Theorem 6.

Corollary 2. *For any constants $k \geq 3$ and $\epsilon > 0$,*

$$\log^{(k)} n\text{-}pass \text{ BPL}[\log n \log^{(k+3)} n] \subseteq \text{DSPACE}(\log n(\log\log n)^{\frac{1}{2}+\epsilon}) \subseteq \text{DTIME}(n^{(\log\log n)^{1/2+\epsilon}}).$$

3.3 Deterministic Simulation of Multi-pass Machines with Linear Advice

Fortnow and Klivans [7] showed that standard (one-pass) randomized log-space machines (BPL) can be simulated by deterministic log-space machines that have access to a linear amount of advice. I.e., they showed that BPL is a subset of L$/O(n)$. On the other hand, using Adleman's technique [1], it can be shown that randomized log-space machines with two-way access to the random tape can be simulated in deterministic log-space using a polynomial amount of advice [16]. Thus, any multi-pass, randomized log-space machine can be simulated in deterministic log-space with polynomial amount of advice. Can we bring down the advice to linear? We show that this is indeed possible with a small increase in space.

Let M be a $O(\log n)$-pass, randomized log-space machine. By Theorem 4, M can be simulated by a one-pass randomized machine that uses $O(\log^2 n)$ space, and by applying the techniques of Fortnow and Klivans [7], it follows that M can be simulated in deterministic space $O(\log^2 n)$ with linear advice. Below we show that we can improve the space bound of the deterministic machine to $O(\log n \log\log n)$. More formally, we prove the following.

Theorem 7. *For any $k(n) \in \omega(1)$, a $k(n)$-pass $s(n)$-space bounded randomized machine using $R(n) = 2^{r(n)}$ random bits can be simulated by a deterministic machine using $O(s(n) + r(n)\log(k(n)s(n)))$ space that uses an advice of size $O(r(n)k(n)s(n) + n)$. I.e.,*

$$k(n)\text{-}pass\ \mathrm{BPSPACE}[s(n), 2^{r(n)}] \subseteq \mathrm{DSPACE}(s(n) + r(n)\log k(n)s(n))/O(r(n)k(n)s(n) + n).$$

Before we sketch a proof of the theorem, we note the following corollaries.

Corollary 3. *For every constant $k > 0$,*

$$O(\log^k n)\text{-}pass\ \mathrm{BPL} \subseteq \mathrm{DSPACE}(\log n \log\log n)/O(n).$$

Corollary 4. *For every $0 < \epsilon < 1$, n^ϵ-pass* BPL $\subseteq$ DSPACE$(\log^2 n)/O(n)$.

Proof sketch of Theorem 7. Consider a $k(n)$-pass $s(n)$-space bounded one-sided randomized machine M that uses $R(n) = 2^{r(n)}$ random bits. By Theorem 5, any pseudorandom generator that fools standard (one-pass) $O(k(n)s(n))$ space-bounded machine using $2^{r(n)}$ random bits also fools M. For our proof, we use Nisan's generator [15]. Note that Nisan's generator, for our choice of parameters of space and random bits, stretches a seed of length $\ell(n) = O(r(n)k(n)s(n))$ to $2^{r(n)}$. The seed for this pseudorandom generator is a tuple consisting of $r(n)$ hash functions from $\{0,1\}^{O(s(n)k(n))}$ to $\{0,1\}^{O(s(n)k(n))}$ and one string of length $O(s(n)k(n))$. Let us denote the hash functions by $h_1, \cdots, h_{r(n)}$ and the string by y. Consider the following simulation M' of M: M' has a seed of length $\ell(n)$ on its random tape. On any input of length n, it will apply Nisan's generator on the contents of random tape and simulate M on the output of the generator. Note that M' accesses the contents of the random tape in a 2-way manner.

By applying Adleman's technique to M', we can fix n seeds which will act as good random string for all strings of length n. We can hardwire the n seeds (each of length $\ell(n)$) and obtain a deterministic simulation. Note that the space used by this simulation is $O(s(n)k(n))$ as we need $O(s(n)k(n))$ space to simulate Nisan's generator and the length of the advice is $O(n\ell(n))$. However, we observe that for any i, the i^{th} bit of the output of the generator can in fact be computed in space $O(r(n) \times \log(k(n)s(n))$. This is because, the output of Nisan's generator is a concatenation of blocks of strings where each block is of length $O(k(n)s(n))$. Each block is obtained by composing $r(n)$ hash functions (in some predetermined order based on the index of the block) on the input y. Note that the output of each individual hash function can be computed in $O(\log(s(n)k(n))$ space. Thus, composition of $r(n)$ hash functions can be computed in $O(r(n)\log(s(n)k(n))$ space. So we get that a $k(n)$-pass $s(n)$-space-bounded randomized machine using $R(n) = 2^{r(n)}$ random bits can be simulated deterministically in space $O(s(n) + r(n)\log(k(n)s(n))$ with $O(n \times \ell(n))$ length advice. Instead of using n independent seeds of length $\ell(n)$, we do a random walk on an expander graph of size $O(2^{\ell(n)})$ and reduce the advice size to $O(n + \ell(n))$, as done by Fortnow and Klivans [7] (using the work of Gutfreund and Viola [9]).

The proof can be extended to two-sided error multi-pass Turing machines as well, as mentioned in [7]. We omit the details. $\square$

4 Conclusions

This paper establishes that time efficient derandomization of probabilistic, log-space machines that make a non-constant passes over the random tape yields a new non-trivial derandomization of probabilistic time. This result suggests that it is fruitful to further study multi-pass, probabilistic, space-bounded machines. One interesting question that arises is on error reduction. Let M be a $k(n)$-pass, $s(n)$-space bounded, bounded-error probabilistic space-bounded machine with error probability less than $1/3$. Can we reduce the error probability to $1/2^{e(n)}$ for a polynomial e without substantial increase in passes and space used? Note that by increasing the number of passes to $O(k(n)e(n))$ this is indeed possible. On the other hand, if we were not to increase the number of passes, then by increasing the space to $O(e(n)s(n))$ we can achieve the same reduction in error probability. Can we do better? Can we reduce the error probability to $1/2^{e(n)}$ while keeping the number of passes to $O(k(n))$ and the space bound to $O(s(n))$?

References

1. L.M. Adleman. Two theorems on random polynomial time. In *Proc. 19th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 75–83, 1978.
2. S. Arora and B. Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, 2009.

3. A. Borodin, S. Cook, and N. Pippenger. Parallel Computation for Well-Endowed Rings and Space-Bounded Probabilistic Machines. *Information and Control*, 58(1-3):113–136, 1983.
4. K.M. Chung, O. Reingold, and S. Vadhan. S-T connectivity on digraphs with a known stationary distribution. *ACM Transactions on Algorithms*, 7(3):30, 2011.
5. M. David, P. Nguyen, P.A. Papakonstantinou, and A. Sidiropoulos. Computationally Limited Randomness. In *Proc. Innovations in Theoretical Computer Science (ITCS)*, 2011.
6. M. David, P.A. Papakonstantinou, and A. Sidiropoulos. How strong is Nisan's pseudo-random generator? *Information Processing Letters*, 111(16):804–808, 2011.
7. L. Fortnow and A.R. Klivans. Linear advice for randomized logarithmic space. In *Proc. Symposium on Theoretical Aspects of Computer Science (STACS)*, 2006.
8. J. Gill. Computational complexity of probabilistic Turing machines. *SIAM Journal on Computing*, 6(4):675–695, 1977.
9. D. Gutfreund and E. Viola. Fooling parity tests with parity gates. In *Proc. APPROX and RANDOM*, pages 381–392. Springer-Verlag, 2004.
10. J.H. Hopcroft, W.J. Paul, and L.G. Valiant. On Time Versus Space. *Journal of the ACM*, 24(2):332–337, April 1977.
11. R. Impagliazzo, N. Nisan, and A. Wigderson. Pseudorandomness for Network Algorithms. In *Proc. 26th ACM Symposium on Theory of Computing (STOC)*, pages 356–364, 1994.
12. G. Karakostas, R.J. Lipton, and A. Viglas. On the complexity of intersecting finite state automata and NL versus NP. *Theoretical Computer Science*, 302:257–274, 2003.
13. M. Karpinski and R. Verbeek. There Is No Polynomial Deterministic Space Simulation of Probabilistic Space with a Two-Way Random-Tape Generator. *Information and Control*, 67(1985):158–162, 1985.
14. R.J. Lipton and A. Viglas. Non-uniform Depth of Polynomial Time and Space Simulations. In *International Symposium on Fundamentals of Computation Theory (FCT)*, pages 311–320. Springer, 2003.
15. N. Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.
16. N. Nisan. On read once vs. multiple access to randomness in logspace. *Theoretical Computer Science*, 107(1):135–144, 1993.
17. R. Raz and O. Reingold. On recycling the randomness of states in space bounded computation. In *Proc. 31st ACM Symposium on Theory of Computing (STOC)*, pages 159–168, 1999.
18. O. Reingold. Undirected connectivity in log-space. *Journal of the ACM*, 55(4):1–24, 2008.
19. O. Reingold, L. Trevisan, and S. Vadhan. Pseudorandom walks on regular digraphs and the RL vs. L problem. In *Proc. 38th ACM Symposium on Theory of Computing (STOC)*, page 457, 2006.
20. M. Saks. Randomization and Derandomization in Space-Bounded Computation. In *Proc. 11th IEEE Conference on Computational Complexity*, 1996.
21. M. Saks and S. Zhou. BPSPACE(S) $\subseteq$ DSPACE($S^{3/2}$). *Journal of Computer and System Sciences*, 403:376–403, 1999.
22. R. Santhanam and D. van Melkebeek. Holographic proofs and derandomization. *SIAM Journal on Computing*, 35(1):59–90, 2005.

A Appendix

Proof of Claim 1. We are assuming that the graph G_M is known. Now the pebbling strategy places a pebble on a vertex i only when all its predecessors have pebbles on them. Thus, by the above observation, we can compute $C(i)$ when a pebble is placed on vertex i. Note that $C(i)$ can be described using $\ell b(n)$ bits. Since for each pebble we store $C(i)$ (for some i), the total space used is the number of pebbles times $\ell b(n)$. Now the number of pebbles used is $O(a(n)/\log a(n))$ as the size of the graph is $a(n)$. So the total space used by the simulating machine is $O\left(a(n)b(n)/\log a(n)\right)$ plus the space needed by the pebbling strategy S. Since the pebbling strategy runs in $O(2^{a^2(n)})$ time, the space used by the pebbling strategy is bounded by $O(2^{a^2(n)})$.

Note that we can reuse the space for each graph, and thus the total space remains same even when we remove our assumption. □

Proof of Claim 2. As before, we are assuming that the graph G_M is known. Even though M reads the contents of the random tape in a one-way manner, the simulating machine may have to read some bits multiple times. Consider a block i on the random tape, the simulating machine needs to access the contents of that block each time the pebbling algorithm S places a pebble on vertex i. If block i is to the left of the current tape head position (of the random tape), then the simulating machine can make a pass over the random tape and access block i. Since the pebbling algorithm takes time $O(2^{a^2(n)})$, the total number of times a pebble is placed on any vertex is at most $O(2^{a^2(n)})$. Thus the simulating machine makes at most $O(2^{a^2(n)})$ passes over the random tape. □

Proof of Observation 2. We can discover whether a graph G is the correct computation graph or not as follows. Observe that given any possible computation graph G, we can compute for each tape the block number the tape-head visits after each time-segment. This information can be stored using $O(a(n)\log a(n))$ bits. Also note that any correct computation graph must have edges of the form $(i, i+1)$, $1 \leq i < a(n)$. Thus, a pebble cannot be placed on vertex $(i+1)$ before placing a pebble on vertex i. This ensures that we will simulate $(i+1)$st time segment only after ith time segment is simulated. Now if G is an incorrect computation graph, then the following scenario happens: The tape-head positions computed based on G will claim that after time-segment i, tape head h is in block b; however, while simulating M, this does not happen. At this point, we will discover that G is not a correct computation graph. □

Proof of Lemma 1. Let M be a bounded-error, probabilistic, $k(n)$-pass, $s(n)$-space-bounded machine that accepts L using $r(n)$ random bits. Consider the following machine N:

1. Input x, $|x| = n$.
2. Let C_0 be the initial configuration of M on input x.
3. Uniformly at random pick $k(n)-1$ many configurations $C_1, C_2, \cdots, C_{k(n)-1}$, and store the configurations on the work tape.

4. For $0 \leq i \leq k(n) - 1$, in parallel DO
   – Simulate one pass of M with C_i as starting configuration.
   – Let C'_i be the configuration after this pass.
5. If there exists an i, $0 \leq i \leq k(n) - 2$, such that $C'_i \neq C_{i+1}$, then accept with probability 1/2 and reject with probability 1/2.
6. Otherwise, accept x if and only if $C'_{k(n)-1}$ is an accepting configuration.

Since each configuration of M is of length $O(s(n))$, the space needed to store all of C_i's is $O(k(n)s(n))$. Clearly, Step 4 can be done by accessing the random-tape in a one-way manner. The total space needed for all $k(n)$ simulations in Step 4 is $O(k(n)s(n))$. Thus N is an $O(k(n)s(n))$-space bounded machine that reads the random tape in a one-way manner. Note that the number of random bits used by the above machine is the number of random bits used in Step 3 (which is $O(k(n)s(n))$ plus the number of random bits used by M. Thus N uses $O(r(n) + k(n)s(n))$ random bits.

Now we bound the success probability of N. Let $x \in L$. For a fixed random string r, let $D^r_1, D^r_2, \cdots, D^r_{k(n)-1}$ be the configurations of $M(x)$ after each pass when r is written on the random tape. Consider the behavior of N when r is written on its random tape. Note that the behavior of $N(x)$ coincides with the behavior of $M(x)$ when for every $1 \leq i \leq k(n) - 2$, C_i equals D^r_i. The probability of this event happening is exactly $1/2^{O(k(n)s(n))}$. When this event does not happen, N accepts x with probability 1/2. Thus

$$\begin{aligned}
&\Pr[N \text{ accepts } x] \\
&= \frac{1}{2^{r(n)}} \sum_{r \in \Sigma^{r(n)}} \Pr[\ N \text{ accepts } x \text{ with } r \text{ on random tape}] \\
&= \frac{1}{2^{r(n)}} \sum_{r \in \Sigma^{r(n)}} \Pr[\ N \text{ accepts } x \mid \forall i C_i = D^r_i] \cdot \Pr[\forall i C_i = D^r_i] \\
&\qquad\qquad + \Pr[N \text{ accepts } x \mid \exists i C_i \neq D^r_i] \cdot \Pr[\exists i, C_i \neq D^r_i] \\
&= \frac{1}{2^{r(n)}} \sum_{r \in \Sigma^{r(n)}} \Pr[M \text{ accepts } x \text{ with } r \text{ on random tape}] \cdot \frac{1}{2^{O(k(n)s(n))}} \\
&\qquad\qquad + \frac{1}{2}\left(1 - \frac{1}{2^{O(k(n)s(n))}}\right) \\
&= \frac{1}{2^{O(k(n)s(n))}} \Pr[M \text{ accepts } x] + \frac{1}{2}\left(1 - \frac{1}{2^{O(k(n)s(n))}}\right) \\
&\geq \frac{1}{2} + \frac{1}{6 \cdot 2^{O(k(n)s(n))}}
\end{aligned}$$

This completes the proof. □