Pseudorandom Walks on Regular Digraphs and the RL vs. L Problem*

Omer Reingold†
Dept. of Computer Science
Weizmann Institute of Science
Rehovot, Israel
omer.reingold@weizmann.ac.il

Luca Trevisan‡
Computer Science Division
U.C. Berkeley
luca@cs.berkeley.edu

Salil Vadhan§
DEAS
Harvard University
Cambridge, MA, USA
salil@eecs.harvard.edu

ABSTRACT

We revisit the general **RL** vs. **L** question, obtaining the following results.

1. Generalizing Reingold's techniques to directed graphs, we present a deterministic, log-space algorithm that given a *regular* directed graph G (or, more generally, a digraph with Eulerian connected components) and two vertices s and t, finds a path between s and t if one exists.

2. If we restrict ourselves to directed graphs that are regular and *consistently labelled*, then we are able to produce *pseudorandom walks* for such graphs in logarithmic space (this result already found an independent application).

3. We prove that if (2) could be generalized to all regular directed graphs (including ones that are not consistently labelled) then $\mathbf{L} = \mathbf{RL}$. We do so by exhibiting a new complete promise problem for **RL**, and showing that such a problem can be solved in deterministic logarithmic space given a log-space pseudorandom walk generator for regular directed graphs.

Categories and Subject Descriptors

F.1.3 [**Computation by abstract devices**]: Complexity Measures and Classes

*Work supported by US-Israel Binational Science Foundation Grant 2002246. A preliminary full version of this paper is posted on ECCC [21].

†Incumbent of the Walter and Elise Haas Career Development Chair

‡Also supported by NSF grant CCF 0515231

§Also supported by NSF grant CCR-0133096, ONR grant N00014-04-1-0478, and a Sloan Research Fellowship

STOC'06, May 21–23, 2006, Seattle, Washington, USA.

General Terms

Algorithms, Theory

Keywords

Expander Graphs, Zig-Zag Product, Space-Bounded Computation, Universal Traversal Sequence, Mixing Time, Derandomization

1. INTRODUCTION

The research on derandomization of space-bounded computations deals with the tradeoff between two basic resources of computations: memory (or space) and randomness. Can randomness save space in computations? Alternatively, can every randomized algorithm be derandomized with only a small increase in space? These questions received the most attention in the context of log-space computations, and with respect to the following complexity classes: **L** (the class of problems solvable in deterministic log-space), **RL**, and **BPL** (the classes of problems solvable by randomized log-space algorithms making one-sided and two-sided errors respectively). It is widely believed that $\mathbf{L} = \mathbf{RL} = \mathbf{BPL}$ and proving this conjecture is the ultimate goal of this body of research.

It turns out that the derandomization of **RL** is related to determining the space complexity of one of the most basic graph problems, Undirected S-T Connectivity: Given an undirected graph and two vertices, is there a path between the vertices? (The corresponding search problem is to find such a path). The space complexity of this problem and the derandomization of space-bounded computations have been the focus of a vast body of work, and brought about some of the most beautiful results in complexity theory. The connection between the two was made by Aleliunas et. al. [2], who gave an **RL** algorithm for Undirected S-T Connectivity. The algorithm simply runs a random walk from the first vertex s for a polynomial number of steps, and accepts if and only if the walk visits the second vertex t. This beautifully simple algorithm is undoubtedly one of the most interesting examples of the potential power of randomization. It casts the space complexity of Undirected S-T Connectivity as a specific example and an interesting test case for the derandomization of space-bounded computations. (In particular, if $\mathbf{RL} = \mathbf{L}$, then Undirected S-T Connectivity can be solved in deterministic log-space.) Since then progress on the general and the specific problems alternated with a fluid exchange of ideas (as demonstrated

by [26, 1, 4, 17, 16, 18, 25, 3], to mention just a few highlights of this research). See the surveys of Saks [24] and Wigderson [29] for more on these vibrant research areas.

The starting point of our research is a recent result of Reingold [20] that showed that UNDIRECTED S-T CONNECTIVITY has a deterministic log-space algorithm. On the other hand, the best deterministic space bound on **RL** in general remains $O(\log^{3/2} n)$, established by Saks and Zhou [25].

1.1 Our Results

In this paper, we revisit the general **RL** vs. **L** question in light of Reingold's results, and obtain the following results:

1. Generalizing Reingold's techniques to directed graphs (aka. *digraphs*), we present a deterministic, log-space algorithm that a graph such that each vertex has an outdegree equal to its indegree (i.e. a digraph with Eulerian connected components) and two vertices s and t, finds a path between s and t if one exists. This involves a new analysis of the zig-zag graph product of [22] that generalizes to regular digraphs, which may be of independent interest.

2. For the special case of "consistently labelled" regular digraphs we provide a "pseudorandom walk generator." A digraph is *regular* of degree D, or D-regular, if all vertices have indegree D and outdegree D; a D-regular digraph is *consistently labelled* if the D edges leaving each vertex are numbered from 1 to D in such a way that at each vertex, the labels of the incoming edges are all distinct. Roughly speaking, given a random seed of logarithmic length, our generator constructs, in log-space, a "short" pseudorandom walk that ends at an almost-uniformly distributed vertex when taken in any consistently-labelled regular digraph.

   Pseudorandom generators that fool space-bounded computations, are very interesting tools even beyond the **RL** vs. **L** problem (see [9, 28, 8, 7] for just a few of their applications). In particular, even the pseudorandom walks given in this paper already found an application in the construction of almost k-wise independent permutations [10]. Unfortunately, such "oblivious" derandomizations are more difficult, due to the inability to look at the input. For example, while it is true that every regular digraph has a consistent labelling, it is not clear how to transform a pseudorandom generator that works for consistently-labelled regular digraphs such that it would also work for arbitrarily-labelled regular digraphs.

3. We prove that if our pseudorandom generator from Item 2 could be generalized to all regular digraphs (instead of just consistently labelled ones), then $\mathbf{RL} = \mathbf{L}$.

   We do so by exhibiting a new complete problem for **RL**: S-T CONNECTIVITY restricted to digraphs for which the random walk is promised to have polynomial "mixing time," as measured by a directed analogue of the spectral gap introduced by Mihail [15]. We then show that a pseudorandom walk generator for regular digraphs can be used to solve our complete problem deterministically in logarithmic space.

1.2 Perspective

We now discuss possible interpretations of the aforementioned results for the derandomization of **RL**.

First, let us consider Reingold's algorithm for undirected ST-connectivity. What are the properties of undirected graphs that are intrinsic to this algorithm? One property of undirected graphs is reversibility — a walk on the graph can immediately undo any of its steps by taking the last edge again (in the reverse direction). A second property is that the stationary distribution of the walk on an undirected graph is well-behaved (the probability of a vertex is proportional to its degree), and such graphs can easily be reduced to regular graphs where the stationary distribution is uniform. Our result (1), where we extend the algorithm to Eulerian digraphs indicates that the latter property of undirected graphs is much more important here than the former. After all, Eulerian digraphs are non-reversible but their stationary distribution is well-behaved and they can easily be reduced to regular digraphs where the stationary distribution is again uniform (the reduction is described in Section 5).

A "pseudorandom walk generator" that works for every consistently labelled regular *undirected* graph is implicit in [20]. (Actually, the generator requires a more restrictive form of labelling). Our result (2) formalizes this generator and generalizes it to (consistently-labelled regular) digraphs. In order to get a general pseudo-random generator for space-bounded generators (which as mentioned above is a goal of independent interest) there are two restrictions to overcome: regularity and consistency of the labelling.

It is well known that every regular digraph has a consistent labelling . Furthermore, regularity already proved crucial in our result (1). It may therefore seem that the most stringent of the requirements in our construction is regularity rather than consistent labelling. Our final result (3) shows that in this context (of derandomization with pseudorandom walks) regularity is essentially irrelevant. Consistent labelling is in fact the only obstacle towards a full derandomization of **RL**. It remains to be seen how difficult this is to overcome.

Why is consistent labelling so important? First, as we noted above, *in the context of pseudorandom walks* it is not clear how useful is the mere fact that consistent labelling *exists*. A pseudorandom walk is an operation that is oblivious to the particular input graph, but on the other hand, consistently labelling a graph may not be oblivious (and in fact seems rather "global"). Therefore, it is not clear how to transform a pseudorandom walk for regular consistently-labelled digraphs into one that is pseudorandom for general regular digraphs. An intuitive reason for the importance of the labelling is that for any fixed sequence of edge labels, the corresponding walk on a graph with consistent labels cannot lose entropy (the distribution of the final vertex has as much entropy as the distribution of the start vertex). On the other hand, without the assumption on the labelling, entropy losses may occur. Therefore progress made in one part of a pseudorandom walk (i.e. an increase in entropy) may be lost later in the same walk.

1.3 Techniques

The main technical step in the proof of our results (1) and (2) is an analysis of a zig-zag graph product of Reingold, Vadhan, and Wigderson [22] applied to regular di-

graphs. More specifically, we bound the *spectral gap* (as defined by Mihail [15] and Fill [6] in the context of nonreversible Markov chains) of the graph obtained by the zig-zag product of two regular digraphs. An analogous bound was proven in [22] for undirected regular graphs, but their proof is not immediately applicable to our setting because it uses properties of symmetric matrices. It turns out that our new analysis is actually simpler than the one in [22], even though it applies to a more general setting. The proof we present here is even simpler than the one that appeared in the preliminary version of this paper [21]. The new proof is based on an approach of [23], who used it to analyze a new 'derandomized squaring' operation.

Another contribution, that may be of independent interest is the new complete promise problem we present for **RL**. Very loosely, this problem is st-connectivity in rapidly mixing Markov chains (where in the 'Yes' case, both nodes s and t have noticeable probability mass under the stationary distribution of walks starting at s). A complete problem for **RL** based on Markov chains was previously known (see the survey of Saks [24]). However, in that problem one examines the behavior of a walk at a particular time step t. On the other hand, in the new complete problem we discuss the behavior of the walk *in its limit* (i.e., we are interested in the stationary distribution). Such a problem seems much more amenable to the techniques of [20]. In particular, even in the undirected case, we do not know how to space-efficiently and deterministically simulate the distribution reached by a random walk after a fixed number of steps (unless this walk was long enough to approach the stationary distribution).

In the proof of our result (3), we define (as a mental experiment) a regular digraph which can be thought of as a "blow-up" of the input graph in the new complete promise problem for **RL**. More specifically, every vertex in the input graph corresponds to a set of vertices in the blow-up graphs, with multiplicity that is linearly related to the weight of the original vertex under the stationary distribution. Intuitively, as heavy vertices are split into many more vertices in the blow-up graph, we indeed obtain a graph where the stationary distribution is uniform (and is therefore regular). We are not able to construct this blow-up graph efficiently but we can show (again as a mental experiment) that for some (inconsistent) labelling of the edges in the blow up graph a walk on the blow-up graph naturally "projects" onto the original graph. Furthermore, the projected walk can be easily and efficiently simulated by only referring to the original input graph. By assumption, we know how to generate pseudorandom walks for the blow-up graph and as we show, simulating the projection of such walks on the original graph is sufficient to solve the promise problem.

It is natural to attempt the general framework of derandomization studied here with a different measure of expansion (rather than analogues of eigenvalue gap). We also consider here the combinatorial measure of *edge expansion*. We show that edge expansion is preserved and degree is reduced, by taking a *replacement product* with an expander graph. We show, however, that edge expansion is not necessarily improved by powering in digraphs, and it is not clear that there is any other "local" operation that increases edge expansion. Details are omitted from this version.

2. PRELIMINARIES

2.1 Graphs and Markov Chains

In this paper, we consider **directed graphs** (**digraphs** for short), and allow them to have multiple edges, and have self-loops. A graph is **out-regular** (resp., **in-regular**) if every vertex has the same number D of edges leaving it; D is called the **out-degree** (resp., **in-degree**). A graph is **regular** if it is both out-regular and in-regular.

Given a graph G on N vertices, we consider the random walk on G described by the transition matrix M_G whose (v,u)'th entry equals the number of edges from u to v, divided by the outdegree of v.[1]

More generally, if $M^{N\times N}$ is a matrix with non-negative entries such that for every $u \in [N]$ we have $\sum_v M(v,u) = 1$, then we say that M is a **Markov chain** on state space $[N]$. For a Markov chain $M^{N\times N}$, we define the **underlying graph** of M as the graph $G = ([N], E)$ such that $(u,v) \in E$ if and only if $M(v,u) > 0$. A distribution $\pi \in \mathbb{R}^N$ is **stationary** for a Markov chain M if $M\pi = \pi$. Note that if π is stationary for M, then $\mathrm{supp}(\pi) \stackrel{\mathrm{def}}{=} \{v : \pi(v) > 0\}$ is a closed subset of M in the sense that there are no transitions from $\mathrm{supp}(\pi)$ to its complement; thus M is well-defined as a Markov chain restricted to $\mathrm{supp}(\pi)$. A Markov chain M is **time reversible** with respect to a stationary distribution π if for every two vertices $u, v \in [N]$ we have $\pi(u)M(v,u) = \pi(v)M(u,v)$. If G is an undirected graph, then M_G is time reversible with respect to the stationary distribution $\pi(u) = d(u)/2m$, where $d(u)$ is the degree of u and m is the number of edges. A random walk on a directed graph, however, is typically not time reversible.

We are interested in the rate at which a Markov chain M converge to a stationary distribution For a time-reversible Markov chain M, it is well-known that the rate of convergence is characterized by the second largest (in absolute value) eigenvalue $\lambda_2(M)$ of the matrix M. If M is not time-reversible (for example, if M is the random walk on a directed graph), then M need not have real eigenvalues, and the stationary distribution need not have the largest eigenvalue in absolute value, so the time-reversible theory is not immediately applicable.

Following Mihail [15] and Fill [6], we introduce a parameter $\lambda(M)$ which is equal to $\lambda_2(M)$ if M is time-reversible, but that remains well-defined even for non-time-reversible Markov chain. For a probability distribution $\pi \in \mathbb{R}^N$ on vertices, we define a normalized inner product on $\mathbb{R}^N$ by:

$$\langle x, y\rangle_\pi \stackrel{\mathrm{def}}{=} \sum_{v\in\mathrm{supp}(\pi)} \frac{x(v)\cdot y(v)}{\pi(v)},$$

and a norm $\|x\|_\pi \stackrel{\mathrm{def}}{=} \sqrt{\langle x, x\rangle_\pi}$. Note that this normalization makes π itself a unit vector (i.e. $\|\pi\|_\pi = 1$), and also implies that x is orthogonal to π iff $\sum_v x(v) = 0$. (Technically, $\langle\cdot,\cdot,\rangle_\pi$ is only an inner product on the subspace $\{x \in \mathbb{R}^N : \mathrm{supp}(x) \subseteq \mathrm{supp}(\pi)\}$, since there are nonzero vectors x outside this subspace such that $\|x\|_\pi = 0$. However, it will be convenient to use this notation for arbitrary vectors in $\mathbb{R}^N$.)

[1]Often the transition matrix is defined to be the transpose of our definition. Our choice means taking a random walk corresponds to *left*-multiplication by M_G.

DEFINITION 2.1. *Let M be a Markov chain and π be a stationary distribution for M. We define the* **spectral expansion** *of M with respect to π to be*

$$\lambda_\pi(M) \stackrel{\text{def}}{=} \max_{x \in \mathbb{R}^N : \langle x, \pi \rangle_\pi = 0} \frac{\|Mx\|_\pi}{\|x\|_\pi},$$

For a digraph G and a stationary distribution of M_G, we often write $\lambda_\pi(G)$ instead of $\lambda_\pi(M_G)$.

As noted above, when M is time-reversible, then $\lambda_\pi(M)$ equals the second largest eigenvalue (in absolute value) of M (more precisely, the submatrix of M consisting of the rows and columns in $\text{supp}(\pi)$). In general, $\lambda_\pi(M)$ equals the square root of the second largest (in absolute value) of $\tilde{M}M$, where $\tilde{M}(u,v) = \pi(u)M(v,u)/\pi(v)$ (again, restricting to submatrices so that $u, v \in \text{supp}(\pi)$).

The following lemma shows that if $\lambda_\pi(M)$ is small, then the Markov chain converges quickly to π. The proof is omitted.

LEMMA 2.2. *Let π be a stationary distribution of Markov chain M on $[N]$, and let α be any distribution on $[N]$ such that* $\text{supp}(\alpha) \subseteq \text{supp}(\pi)$. *Then*

$$\|M^t\alpha - \pi\|_\pi \leq \lambda_\pi(M)^t \cdot \|\alpha - \pi\|_\pi.$$

In particular, if we start at a vertex $v \in \text{supp}(\pi)$ and run M for t steps, then we end at vertex $w \in \text{supp}(\pi)$ with probability at least $\pi(w) - \lambda_\pi(M)^t \cdot \sqrt{\pi(w)/\pi(v)}$.

The above lemma refers to convergence in (normalized) ℓ_2 distance. The following lemma shows that this implies convergence in standard variation distance. The proof is omitted.

LEMMA 2.3. *For any distribution α, the variation distance between α and π is at most $\|\alpha - \pi\|_\pi$.*

It is well-known that (connected, nonbipartite) undirected graphs G always satisfy $\lambda_\pi(G) \leq 1 - 1/\text{poly}(N, D)$, where N is the number of vertices and D the degree [12]. That is, undirected graphs have at most polynomial mixing time. However, in general directed graphs, $\lambda_\pi(G)$ can be exponentially close to 1, and thus the mixing time exponentially large.

Just as in the undirected case, the spectral expansion can be bounded in terms of the sizes of cuts in the underlying graph.

DEFINITION 2.4. *Let M be a Markov chain with N vertices and π a stationary distribution. The* **conductance** *of M with respect to π is defined to be*

$$h_\pi(M) \stackrel{\text{def}}{=} \min_{A : 0 < \pi(A) \leq 1/2} \frac{\sum_{u \in A, v \notin A} \pi(u)M(v,u)}{\pi(A)}.$$

LEMMA 2.5 ([27, 15, 6]). *Let M be a Markov chain on N vertices such that $M(u,u) \geq 1/2$ for every u (i.e. M is "strongly aperiodic"), and let π be a stationary distribution of M. Then $\lambda_\pi(M) \leq 1 - h_\pi(M)^2/2$.*

The proof is omitted.

When the stationary distribution π is uniform on the vertices of G, then the conductance defined above coincides exactly with the "edge expansion" of G, defined below.[2]

[2]To see that $\varepsilon(G) = h_\pi(G)$ when π is the uniform distribution, note that the fact that the stationary distribution is uniform implies that G is biregular, which in turn implies that $E(A, \overline{A}) = E(\overline{A}, A)$.

DEFINITION 2.6. *Let $G = (V, E)$ be a directed graph in which every vertex has outdegree D. Then the* **edge expansion** *of G is defined to be*

$$\varepsilon(G) = \min_A \frac{E(A, \overline{A})}{D \cdot \min\{|A|, |\overline{A}|\}},$$

where the minimum is taken over sets of vertices A and $E(A, \overline{A})$ is the set of edges (u, v) where $u \in A$ and $v \notin A$.

2.2 Complexity Classes

We let **L**, **RL**, **NL**, **BPL** denote the standard logspace complexity classes. That is, **L** is the class of decision problems solvable by *deterministic* logarithmic space Turing machines, **RL** is the class of decision problems solvable by *probabilistic* logarithmic space Turing machines with bounded one-sided error, **BPL** is the class of decision problems solvable by *probabilistic* logarithmic space Turing machines with bounded two-sided error, and **NL** is the class of decision problems solvable by *non-deterministic* logarithmic space Turing machines. We require our machines to always terminate for every input and for every sequence of random coins or non-deterministic choices. In particular, this implies that every computation terminates within polynomial time.

We also define the *promise version* of log-space complexity classes. A *promise* problem is a pair (Y, N) of disjoint sets of instances. A promise problem (Y, N) is in the class **prL** if there is a deterministic log-space Turing machine that accepts all the inputs in Y and rejects all the inputs in N. A promise problem (Y, N) is in **prRL** if there is a probabilistic logarithmic space Turing machine that accepts inputs in Y with probability at least $1/2$ and accepts inputs in N with probability 0. A promise problem (Y, N) is in **prBPL** if there is a probabilistic logarithmic space Turing machine that accepts inputs in Y with probability at least $3/4$ and accepts inputs in N with probability at most $1/4$. When dealing with promise problems, we require probabilistic machines to halt for every input in $Y \cup N$ and for every sequence of random coins. (We allow infinite loops for inputs not in $Y \cup N$.)

Finally, we define complexity classes of *search problems*. A **search problem** is simply a relation $R \subseteq \Sigma^* \times \Sigma^*$. For a relation R and a string x we define $R(x) \stackrel{\text{def}}{=} \{y : R(x, y)\}$. The computational problem associated with a search problem R is the following: given x such that $R(x) \neq \emptyset$, output a string y in $R(x)$.

A relation (or search problem) R is **log-space** if there is a polynomial p such that $y \in R(x)$ implies $|y| \leq p(|x|)$ and if the predicate $(x, y) \in R$ can be decided by a log-space deterministic Turing machine that has two-way access to x and one-way access to y.

A logspace search problem R is in **searchL** if there is a logarithmic space transducer A such that $A(x) \in R(x)$ for every x such that $R(x) \neq \emptyset$. (A transducer is a Turing machine with a read-only input tape, a work tape, and a write-only output tape. The writing head on the output tape is constrained to always move right after writing a symbol, but the machine has two-way access to the input tape.)

A logspace search problem R is in **searchRL** if there is a logarithmic space probabilistic transducer A and a polynomial p such that $\Pr[A(x) \in R(x)] \geq \frac{1}{p(x)}$ for every x such that $R(x) \neq \emptyset$. (We require the transducer to halt for every sequence of random coins and for every x such that $R(x) \neq \emptyset$.)

All **reductions** in this paper are deterministic logspace reductions. The definition of reduction is standard for decision problems and promise problems. The definition for search problems is omitted.

3. A NEW COMPLETE PROBLEM FOR RL

S-T CONNECTIVITY and its search version, FIND PATH, both defined below, are two of the most basic problems in computer science.

S-T CONNECTIVITY.
Input: (G, s, t), where $G = (V, E)$ is a directed graph, $s, t \in V$.
YES instances: There is a path from s to t in G.
NO instances: There is no path from s to t in G.

FIND PATH.
Input: (G, s, t), where $G = (V, E)$ is a directed graph, $s, t \in V$, and $k \in \mathbb{N}$.
Promise: There is a path from s to t in G.
Output: A path from s to t in G.

It is well-known that S-T CONNECTIVITY is complete for **NL**, and the same argument shows that FIND PATH is complete for **searchNL**. Here we are interested in the complexity of restrictions of these problems. The recent result of Reingold [20] shows that their restrictions to *undirected* graphs, UNDIRECTED S-T CONNECTIVITY and UNDIRECTED FIND PATH, are in **L** and **searchL**, respectively.

It was known (see [24]) that a certain restriction of S-T CONNECTIVITY was complete for **prRL**, specifically one where we look at the probability that a random walk of a particular length goes from s to t:

SHORT-WALK S-T CONNECTIVITY.
Input: $(G, s, t, 1^k)$, where $G = (V, E)$ is a directed graph, $s, t \in V$.
YES instances: A random walk of length k started from s ends at t with probability at least $1/2$.
NO instances: There is no path from s to t in G.

However, this problem does not seem to capture the properties of UNDIRECTED S-T CONNECTIVITY used in Reingold's algorithm [20]. His algorithm uses relies on a measure of expansion, specifically the spectral gap, which refers to the *long-term* behavior of random walks in G (as opposed to walks of a particular length k). We give a complete problem that seems much closer, specifically by restricting to graphs of polynomial mixing time (as measured by $\lambda_\pi(G)$).

POLY-MIXING S-T CONNECTIVITY.
Input: $(G, s, t, 1^k)$, where $G = (V, E)$ is a out-regular directed graph, $s, t \in V$, and $k \in \mathbb{N}$.
YES instances: The random walk on G has a stationary distribution π such that $\lambda_\pi(G) \leq 1 - 1/k$, and $\pi(s), \pi(t) \geq 1/k$.
NO instances: There is no path from s to t in G.

POLY-MIXING FIND PATH.
Input: $(G, s, t, 1^k)$, where $G = (V, E)$ is a out-regular directed graph, $s, t \in V$, and $k \in \mathbb{N}$.
Promise: $\lambda_s(G) \leq 1 - 1/k$, and $\pi_s(s), \pi_s(t) \geq 1/k$.
Output: A path from s to t in G.

The completeness of these two problems is given by the following theorem.

THEOREM 3.1. POLY-MIXING S-T CONNECTIVITY *is complete for* **prRL**. POLY-MIXING FIND PATH *is complete for* **searchRL**.

PROOF. First, we show that these problems are in **prRL** and **searchRL**, respectively, by giving randomized logspace algorithms for them. Given an instance $(G, s, t, 1^k)$, we take a random walk of length $m = 2k \cdot \ln k$ from s, where N is the number of vertices in G. The **searchRL** algorithm simply outputs this walk, and the **prRL** algorithm accepts if this walk ends at t. If $(G, s, t, 1^k)$ is a YES instance, then by Lemma 2.2, the random walk will end at t with probability at least

$$\begin{aligned}\pi(t) - \lambda_\pi(M)^m \cdot \sqrt{\pi(t)/\pi(s)} &\geq \frac{1}{k} - \left(1 - \frac{1}{k}\right)^m \cdot \sqrt{k} \\ &\geq \frac{1}{k} - \frac{1}{k^{3/2}} \geq \frac{1}{2k}.\end{aligned}$$

Now we show that every problem in **prRL** and **searchRL**, respectively, reduce to POLY-MIXING S-T CONNECTIVITY and POLY-MIXING FIND PATH. Let M be a randomized logspace machine, running in time at most $p(n) \leq \text{poly}(n)$. Given an input x of length n for M, we construct a graph G whose vertices are of the form (i, τ), where $i \in \{1, \ldots, p(n)\}$ is a "layer", and $\tau \in \{0,1\}^{O(\log n)}$ describes a possible configuration of M (i.e. the state, the contents of the work tape, and the position of the input head). We let $s = (1, \alpha)$ where α is the unique start configuration of M, and $t = (p(n), \beta)$ where β is the (wlog unique) accepting configuration of M. (In the case of a **searchRL** algorithm, we have M accept if any of the strings it outputs satisfy the relation R.) We create four outgoing edges from each vertex (i, γ). Two edges are always self-loops. If $i = p(n)$, then the other two edges go to s. If $i < p(n)$, then the they go to vertices of the form $(i+1, \gamma')$ and $(i+1, \gamma'')$, for γ', γ'' as follows. If γ is a configuration where M reads a new random bit, then we take γ' and γ'' to be the two configurations that M would enter depending on the two possible values of the random bit. If γ is a configuration where M does not read a new random bit, then we set $\gamma' = \gamma''$ to be the unique next configuration in M's computation on x. If γ is a halting configuration of M, then we set $\gamma' = \gamma'' = \gamma$.

Let us analyze the stationary distribution and mixing time of a random walk on G. It can be verified that the following distribution π is on vertices (i, τ) is stationary for G: choose i uniformly at random from $\{1, \ldots, p(n)\}$, run M for i steps on input x, and let τ be M's configuration. We see that if $x \in L$, then $\pi(t) > 1/2p(n)$, and if $x \notin L$, then $\pi(t) = 0$. In both cases $\pi(s) = 1/p(n)$.

To bound the mixing time, we observe that a random walk of length $3p(n)$ started at *any* vertex visits s with probability $1 - 2^{-\Omega(p(n))} \geq 1/2$. Lemma 3.2 below says that G has a stationary distribution π' such that $\lambda_{\pi'}(G) \leq 1 - 1/(8 \cdot (3p(n))^2)$ and $\pi'(s) > 0$. It follows that π' is the unique stationary distribution on G, since a random walk started at any vertex eventually passes through s and thus converges to π' (by Lemma 2.2). So $\pi' = \pi$.

To conclude, in our reduction, we output $(G, s, t, 1^k)$, where $k = 72p(n)^2$. From the analysis above, this gives a logspace

reduction from any problem in **prRL** to Poly-Mixing S-T Connectivity. Similarly, it gives a reduction from any problem in **searchRL** to Find Path, because with one-way access to any path from s to t in G, in logspace we can construct polynomially many computation paths of M, at least one of which is accepting, and this in turn, can be used to obtain a polynomially many strings $y_1, \ldots, y_\ell$ at least of which is in $R(x)$. ■

The above proof required the following lemma (proof is omitted), which says that to show that a Markov chain has polynomial mixing time, it suffices to prove that there is a vertex s such that a random walk of polynomial length started at any vertex will visit s with high probability.

Lemma 3.2. *Let M be a Markov chain that is strongly aperiodic (i.e. self-loop probability at least 1/2 at each vertex). Suppose there is a vertex s and a number $\ell \in \mathbb{N}$ such that from every vertex v reachable from s, a random walk of length ℓ from v visits s with probability at least $1/2$. Then M has a stationary distribution π such that $\lambda_\pi(G) \leq 1 - 1/8\ell^2$ and $\pi(s) \geq 1/2\ell$.*

In fact, the converse is also true — if a Markov chain has polynomial mixing time then there is a vertex s such that a random walk of polynomial length started at any vertex will visit s with high probability. Indeed, if $\lambda_\pi(M) \leq 1 - \gamma$ and we take s to be any vertex such that $\pi(s) \geq 1/N$ (where N is the number of states), then Lemma 2.2 says that a random walk of length $\ell = O((1/\gamma) \cdot \log(N/p_{min}))$ will end at s with probability at least $1/2N$, where p_{min} is the minimum (nonzero) probability mass under π. Repeating $O(N)$ times, we visit s with high probability. In cases we are interested in (e.g. random walks on graphs), p_{min} is only exponentially small, so the walk length $\ell \cdot N$ is polynomial.

We note that the proof of Theorem 3.1 can be modified to give a complete problem for **prBPL**, specifically where the NO instances are replaced with instances such that $\lambda_\pi(G) \leq 1 - 1/k$, $\pi(s) \geq 1/k$ and $\pi(t) \leq 1/2k$.

4. OPERATIONS ON DIRECTED GRAPHS

Given the **RL**-complete problem from the previous section, it is natural to ask whether Reingold's algorithm [20] for Undirected S-T Connectivity can be generalized to work for the complete problem. Recall that the algorithm works by taking any undirected graph G and applying a sequence of operations to improve its expansion, as measured by spectral gap. Specifically, it relies on a pair of operations that doubles the spectral gap while keeping the degree constant (and increasing the number of vertices by a constant factor). Since the initial (non-bipartite, connected) undirected graph G has spectral gap $\gamma(G) \stackrel{\text{def}}{=} 1 - \lambda(G) \geq 1/\text{poly}(N)$, after $O(\log N)$ operations, we have a graph G' with $\gamma(G') \geq 1/2$. That is, G' is a (constant-degree) expander graph and in particular has diameter $O(\log N)$ (in each connected component). Then s-t connectivity can be decided in logspace by enumerating all paths of $O(\log N)$ from s.

Attempting to generalize this approach to the **RL**-complete problem Poly-Mixing S-T Connectivity, we observe that the initial condition $\gamma(G) \geq 1/\text{poly}(N)$ holds by the promise (taking N to be the length of the input). In addition, if we manage to convert G into a constant-degree graph G' with $\gamma(G') \geq 1/2$ while maintaining the fact that s and t have stationary probability at least $1/\text{poly}(N)$, then Lemma 2.2 implies that there is a path of length $O(\log N)$ from s to t and we can solve s-t connectivity by enumerating all such paths.

Thus, the "only" missing part of the algorithm is generalizing the operations used by Reingold to improve expansion (without increasing the degree) to directed graphs. Below we suggest some possibilities.

Labellings. Let G be a digraph with N vertices such that every vertex has outdegree at most D_{out} and indegree at most D_{in}. (Recall that we allow multiple edges and self-loops.) A *two-way labelling* of G provides a numbering of the edges leaving each vertex of G using some subset of the numbers from 1 to D_{out}, as well as a numbering of edges entering each vertex of G using some subset of the numbers from 1 to D_{in}. (No two edges leaving a vertex can have the same number, and no two edges entering a vertex can have the same number.) Such a graph together with its two-way labelling can be specified by a **rotation map** $\text{Rot}_G : [N] \times [D_{\text{out}}] \to ([N] \times [D_{\text{in}}]) \cup \{\perp\}$, where $\text{Rot}_G(v, i) = (u, j)$ if there is an edge numbered i leaving v and it equals the edge numbered j entering u, and $\text{Rot}_G(v, i) = \perp$ if there is no edge numbered i leaving v. The operations below will be defined in terms of 2-way labellings, as specified by rotation maps.

Operations. The first operation used by Reingold [20] to improve expansion is powering, simply replaces the edge set with all walks of length t in the graph.

Definition 4.1 (powering). *Let G be a two-way labelled graph given by rotation map $\text{Rot}_G : [N] \times [D] \to [N] \times [B]$. The **$t$'th power** of G is the graph G^t with rotation map is given by $\text{Rot}_{G^t} : [N] \times [D]^t \to [N] \times [B]^t$ defined by $\text{Rot}_{G_t}(v_0, (k_1, k_2, \ldots, k_t)) = (v_t, (\ell_t, \ell_{t-1}, \ldots, \ell_1))$, where these values are computed via the rule $(v_i, \ell_i) = \text{Rot}_G(v_{i-1}, k_i)$ (and if any of these evaluations yield $\perp$, then the final output is also $\perp$).*

In directed graphs, powering improves expansion (i.e. reduces mixing time) as well as it does in undirected graphs:

Lemma 4.2. *For any stationary distribution π of G, $\lambda_\pi(G^t) \leq \lambda_\pi(G)^t$.*[3]

Powering alone does not suffice, because it increases the degree of the graph. Thus, Reingold [20] requires an additional operation to reduce the degree while maintaining the expansion. For this, there are two possibilities — the replacement product and zig-zag product. These operations were defined and analyzed by Reingold, Vadhan, and Wigderson [22] for undirected regular graphs, and it is not clear what is the 'right' generalization to irregular directed graphs (particularly non-Eulerian graphs, where the indegree and outdegree of an individual vertex may be unequal). Here we suggest one possibility. For simplicity, we restrict to rotation maps where the outdegree bound D is the same as the indegree bound B.

In the replacement product, we combine a graph G_1 with N_1 vertices and a rotation map of degree D_1 with a graph

[3]In undirected graphs this is actually an equality, but in digraphs it need not be.

G_2 that has D_2 vertices and a rotation map of degree D_2. The product graph has D_1N_1 vertices, that we think of as being grouped into N_1 "clouds" of size D_1, one cloud for each vertex of G_1. Each cloud is a copy of the graph G_2. In addition, if the i-th outgoing edge from vertex v in G_1 was the j-th incoming edge in w (that is, if $\mathrm{Rot}_{G_1}(v,i) = (w,j)$, then, in the product graph, there is an edge from the i-th vertex in the cloud of v to the j-th vertex in the cloud of w. The formal definition follows.

DEFINITION 4.3 (REPLACEMENT PRODUCT). *If G_1 is a two-way labelled graph on N_1 vertices with rotation map $\mathrm{Rot}_{G_1} : [N_1] \times [D_1] \to [N_1] \times [D_1]$ and G_2 is a two-way labelled graph on D_1 vertices with rotation map $\mathrm{Rot}_{G_2} : [D_1] \times [D_2] \to [D_1] \times [D_2]$, then their* **replacement product** *$G_1 ⓡ G_2$ is defined to be the graph on $[N_1]\times[D_1]$ vertices whose rotation map $\mathrm{Rot}_{G_1ⓡG_2} : ([N_1] \times [D_1]) \times [D_2 + 1] \to ([N_1] \times [D_1]) \times [D_2 + 1]$ is as follows:*

$\mathrm{Rot}_{G_1ⓡG_2}((v,k),i)$*:*

1. *If $i \le D_2$, let $(m,j) = \mathrm{Rot}_{G_2}(k,i)$ and output $((v,m),j)$.*
2. *If $i = D_2 + 1$, output $(\mathrm{Rot}_{G_1}(v,k),i)$.*
3. *If the computation of Rot_{G_2} or Rot_{G_1} yields $\perp$, then the output is $\perp$.*

A variant, called the **balanced replacement product** $G_1 ⓑ G_2$ in [22], gives equal weight to the edges coming from G_1 and from G_2, by duplicating edges that go between clouds (ie edges of the type 2) D_2 times, for a total degree of $2D_2$.

The zig-zag product, introduced in [22], combines, as before, a graph G_1 with N_1 vertices and a rotation map of degree D_1 with a graph G_2 that has D_1 vertices and degree D_2. The product graph has N_1D_1 vertices as in the replacement product, but now there is an edge between two vertices if there is a length-three path in the replacement product graph between them, and the middle edge in the path crosses between two clouds. In particular, the degree of the zig-zag product graph is D_2^2, instead of $D_2 + 1$. The formal definition is below.

DEFINITION 4.4 (ZIG-ZAG PRODUCT [22]). *If G_1 is a labelled graph on N_1 vertices with rotation map $\mathrm{Rot}_{G_1} : [N_1]\times[D_1] \to [N_1] \times [D_1]$ and G_2 is a labelled graph on D_1 vertices with rotation map $\mathrm{Rot}_{G_2} : [D_1] \times [D_2] \to [D_1] \times [D_2]$, then their* **zig-zag** *product $G_1 ⓩ G_2$ is defined to be the graph on $[N_1]\times[D_1]$ vertices whose rotation map $\mathrm{Rot}_{G_1ⓩG_2} : ([N_1] \times [D_1]) \times [D_2^2] \to ([N_1] \times [D_1]) \times [D_2^2]$ is as follows:*

$\mathrm{Rot}_{G_1ⓩG_2}((v,k),(i,j))$*:*

1. *Let $(k',i') = \mathrm{Rot}_{G_2}(k,i)$.*
2. *Let $(w,\ell') = \mathrm{Rot}_{G_1}(v,k')$.*
3. *Let $(\ell,j') = \mathrm{Rot}_{G_2}(\ell',j)$.*
4. *Output $((w,\ell),(j',i'))$.*

In typical applications of the zig-zag or replacement products (e.g. [22, 20], G_2 is taken to a constant-degree expander graph (i.e. $\gamma(G_2) = \Omega(1)$). Then, for the case of undirected graphs, it is known that the zig-zag product and the balanced replacement product have spectral gap that is at most a constant factor smaller than the spectral gap of G_1 [22, 14].[4] Thus they roughly maintain expansion while reducing the degree to a constant, and this suffices for Reingold's algorithm [20].

Unfortunately, we do not know how to analyze the effect of the zig-zag and/or replacement products (or variants) on spectral gap for directed graphs in general. Indeed, even the stationary distribution is not well-behaved under these products; we can construct examples where the stationary probability of a vertex t goes from being noticeable (e.g. $1/N^2$) to exponentially small. In the full version we show that the replacement product can actually be analyzed with respect to *edge expansion*, but then it turns out that powering no longer behaves well.

We can analyze these products (and thus extend Reingold's algorithm) for the case of *regular* digraphs, and these results are presented in the next section.

5. REGULAR (AND EULERIAN) GRAPHS

We define REGULAR DIGRAPH S-T CONNECTIVITY and REGULAR DIGRAPH FIND PATH to be the problems obtained by restricting S-T CONNECTIVITY and FIND PATH to regular digraphs, and similarly EULERIAN S-T CONNECTIVITY and EULERIAN FIND PATH to be the restrictions to digraphs where every connected component is Eulerian, i.e. where every vertex has the same in-degree as out-degree. There is no additional promise in these problems. As observed in [11], EULERIAN S-T CONNECTIVITY reduces to UNDIRECTED S-T CONNECTIVITY, simply by making all edges undirected. Whether or not s and t are connected is maintained because, in an Eulerian graph, every cut has the same number of edges crossing in both directions. Note, however, that this is *not* a reduction from EULERIAN FIND PATH to UNDIRECTED FIND PATH. Nevertheless, here we give a logspace algorithm for EULERIAN FIND PATH by generalizing the ideas underlying Reingold's algorithm [20] to the directed case.

THEOREM 5.1. EULERIAN FIND PATH *is in* **searchL**.

Actually, to prove Theorem 5.1, it suffices to provide a logspace algorithm for REGULAR DIGRAPH FIND PATH, because Eulerian digraphs can be reduced to the case of 2-regular digraphs by replacing each vertex v with a directed cycle C_v of $\deg(v)$ vertices, where we connect one outgoing edge of v and one incoming edge of v to each of the vertices in C_v. Thus in the rest of this section we focus on regular digraphs.

5.1 Basic Facts

In a regular digraph of degree D, the rotation map $\mathrm{Rot}_G : [N] \times [D] \to [N] \times [D]$ is a permutation. Note that the uniform distribution is a stationary distribution of the random walk on a regular digraph. Thus, when working with regular digraphs, the inner product $\langle\cdot,\cdot\rangle_\pi$ and the spectral expansion $\lambda_\pi(G)$ will always be with respect to π being the

[4] Actually, the undirected definitions of these products are restricted to two-way labellings that are *undirected* in the sense that every edge $\{u,v\}$ has the same label as an edge leaving u as it does entering u. That is, Rot $\circ$ Rot is the identity.

uniform distribution, and we will usually omit π from the notation.

First, we note that regular digraphs have nonnegligible spectral gap, just like in the undirected case, provided every vertex has a self-loop.[5] The proof is omitted.

LEMMA 5.2. *Let G be a connected, D-regular digraph on N vertices in which every vertex has at least αD self-loops. Then $\lambda(G) \leq 1 - \Omega(\alpha/DN^2)$.*

5.2 Zig-zag Product

In this section, we generalize the Zig-Zag Theorem of [22] to regular digraphs.

THEOREM 5.3. *If $\lambda(G_1) \leq 1 - \gamma_1$ and $\lambda(G_2) \leq 1 - \gamma_2$, then $\lambda(G_1 \textcircled{z} G_2) \leq 1 - \gamma_1 \cdot \gamma_2^2$.*

Our algorithm, like [20], we will only use the following consequence of the second bound above: if G_2 is a good expander in the sense that $\lambda(G_2)$ is bounded by a constant less than 1 and $\lambda(G_1) \leq 1 - \gamma_1$, then $\lambda(G_1 \textcircled{z} G_2) \leq 1 - \Omega(\gamma_1)$. In the preliminary version of this paper [21], we presented a proof of this $1 - \Omega(\gamma_1)$ that was conceptually simpler than the previous proofs of this bound in the undirected case, for either the zig-zag or replacement products.[6] In the full version, we give an even simpler proof, based on an approach of Rozenman and Vadhan [23], who used it to analyze a new 'derandomized squaring' operation (that gives an alternative to Reingold's algorithm as well our generalization to Eulerian digraphs). The proof is omitted.

5.3 The Path-Finding Algorithm

We have seen that powering and the zig-zag graph product has essentially the same affect on regular digraphs as on undirected graphs. Therefore, both the decision and search versions of the st-connectivity algorithm of [20] can be extended (without any substantial change) to regular digraphs. This implies Theorem 5.1, which states that REGULAR DIGRAPH FIND PATH is in **searchL**. The algorithm here is essentially the same as in [20], it is omitted from this version.

6. OBLIVIOUS ALGORITHMS FOR CONSISTENTLY LABELLED GRAPHS

The algorithm given for REGULAR DIGRAPH FIND PATH in the previous section is in the standard computational model, where the input graph is given explicitly to the logspace algorithm. However, for s-t connectivity problems, it is also interesting to seek "oblivious" algorithms that do not explicitly get the input graph, but are only able to walk on the graph by specifying a sequence of outgoing edge labels. That is, the algorithm is given the parameters of the input graph G (namely, number of vertices N and degree D), and then tries to produce a walk $w \in [D]^*$ such that the walk in G obtained starting at s and following the edge labels in w visits t at some point.

Notice that the behavior of such an oblivious algorithm is sensitive to the labelling of outgoing edges in G, but incoming edge labels are irrelevant. Thus, now we think of our D-regular digraph G as being specified with a *one-way labelling*; that is, the outgoing edges from each vertex are numbered from 1 to D. (In contrast, the algorithm presented in the previous section can be thought of as being given an *unlabelled* graph, then it constructs its own two-way labelling to facilitate the applications of the zig-zag product.)

Here we present two types of oblivious algorithms for regular digraphs, one being a deterministic, logspace construction of "universal traversal sequences" and the other being a logspace-computable "pseudorandom generator" for random walks on the graph.

These algorithms will only work on regular digraphs that are **consistently labelled**, which means that all the edges coming into any vertex of the graph have distinct labels, i.e. no vertex v can be both u's i^{th}-neighbor and w's i^{th}-neighbor (for any distinct vertices u and w). In other words, if we use the same labels to number the edges incoming at each vertex (if (u, v) is the i'th edge leaving u, we consider it to be the i'th edge entering v), we obtain a legal *two-way* labelling of the graph (in that each label in $[D]$ will get used exactly once as an incoming label each vertex). Every regular digraph has a consistent labelling; this is equivalent to the fact that every D-regular bipartite graph is the union of D perfect matchings. . However, finding a consistent labelling may not be feasible in log-space, and in any case an oblivious algorithm does not have the freedom to relabel the graph.

We remark that oblivious algorithms like the ones we describe often have applications that non-oblivious algorithms may not. For example, pseudorandom generators fooling logspace algorithms, such as [17, 19], have a variety of applications that do not seem to follow arbitrary deterministic simulations of **RL**, e.g. [9, 28, 8, 7]. Even our pseudorandom generator in Section 6.2 below has already found an application in the construction of almost k-wise independent permutations [10]. All proofs are omitted from this section.

6.1 Universal Traversal Sequences

DEFINITION 6.1 ([2]). *Let D and N be two integers and let $\mathcal{G}$ be a subset of the labelled D-regular connected digraphs on N vertices. We say that a sequence of values in $[D]$ is a* **universal traversal sequence** *for $\mathcal{G}$, if for every graph $G \in \mathcal{G}$, and every vertex $s \in [N]$, the walk that starts in s and follows the edges of G according to the sequence of labels visits all the vertices of the graph.*

We will show how the REGULAR DIGRAPH FIND PATH algorithm described in the previous section also implies a log-space constructible universal traversal sequence for *consistently labelled* regular digraphs.

THEOREM 6.2. *There exists a log-space algorithm that on input $1^N, 1^D$ produces a universal traversal sequence for all connected, consistently labelled D-regular digraphs G on N-vertices.*

[5] In the preliminary version of this paper [21], we erroneously used the standard notion of aperiodicity (i.e. gcd of all cycle lengths is 1) instead of requiring self-loops. However, the lemma is false in this case; see [23].

[6] The basic analysis of the undirected zig-zag product in [22] only gives a bound of $1 - \Omega(\gamma_1^2)$. Only a much more complicated and less intuitive analysis, that uses the undirectedness of G_1 in additional ways, gives the $1 - \Omega(\gamma_1)$ bound. The Martin–Randall [14] decomposition theorem for Markov chains also implies a $1 - \Omega(\gamma_1)$ bound for the undirected replacement products, but its full proof (relying on [5]) is also fairly involved.

6.2 A Pseudorandom Generator

In this section we show that the path finding algorithm also implies a generator with logarithmic seed length that produces in log-space a "pseudorandom walk" for consistently labelled regular digraphs. This means that from any start vertex, following the pseudorandom walk leads to an almost uniformly distributed vertex. In other words, just as the random walk, the pseudorandom walk converges to the stationary distribution. This seems to be a result of independent interest. In particular, we show in Section 7 that a similar pseudorandom generator (or even weaker), that works for regular digraphs with *arbitrary labels*, would prove that $\mathbf{RL} = \mathbf{L}$.

The intuition for the generator is as follows. In the path-finding algorithm, an expander graph G_{exp} is constructed. In this graph a short random walk converges to the uniform distribution. As in the proof for the universal traversal sequences, the sequence of labels of the (random) walk on G_{exp} can be translated to a (pseudorandom) sequence of labels for a walk on G. Furthermore, this sequence of labels is independent of G (and can be computed in log-space without access to G). Note that all nodes of the original graph G are expanded to "clouds" of equal size. Therefore, the pseudorandom walk converges to the uniform distribution on the vertices of G (which is the projection on G of the uniform distribution on the vertices of G_{exp}). Formalizing the above arguments will indeed imply a generator that produces a pseudorandom walk of length polynomial in the size of the graph. However, a truly random walk will converge faster if G has a larger eigenvalue gap. Theorem 6.3 takes this into account and implies, in this case, a pseudorandom walk that is shorter as well.

THEOREM 6.3. *For every $N, D \in \mathbb{N}$, $\delta, \gamma > 0$, there is a generator* $\text{PRG} = \text{PRG}_{N,D,\delta,\gamma} : \{0,1\}^r \to [D]^\ell$ *with seed length $r = O(\log(ND/\delta\gamma))$, and walk length $\ell = \text{poly}(1/\gamma) \cdot \log(ND/\delta)$, computable in space $O(\log(ND/\delta\gamma))$ such that for every* consistently labelled *$(N, D, 1-\gamma)$ regular digraph G and every vertex s in G, talking walk $\text{PRG}(U_r)$ from s ends at a vertex that is distributed δ-close to uniform (in variation distance).*

7. REDUCING ALL OF RL TO THE REGULAR CASE

In this section, we prove that if there exists a pseudorandom generators for walks on regular digraphs *whose edges are arbitrarily labelled*, then $\mathbf{RL} = \mathbf{L}$ and also $\mathbf{searchRL} = \mathbf{searchL}$. Theorem 6.3 implies a generator for walks on regular digraphs with the additional restriction that the labelling of the edges is consistent. Lifting this restriction would imply that $\mathbf{RL} = \mathbf{L}$. In fact, such a generator would also imply $\mathbf{BPL} = \mathbf{L}$. However, we concentrate in this preliminary version on the case of $\mathbf{RL}$.

THEOREM 7.1. *There is a universal constant $\alpha > 0$ such that the following holds for every constant $a \in \mathbb{N}$. Suppose that for every $N, D \in \mathbb{N}$, $\delta, \gamma > 0$, there is a generator* $\text{PRG} = \text{PRG}_{N,D,\delta,\gamma} : \{0,1\}^r \to [D]^\ell$ *with seed length $r = a\log(ND/\delta\gamma)$, and walk length $\ell = (1/(\gamma\delta))^a \cdot (ND)^\alpha$, computable in space $a\log(ND/\delta\gamma)$ such that for every $(N, D, 1-\gamma)$ regular digraph $G = (V, E)$ and every vertex $s \in V$ and every subset $T \subseteq V$ of density at least δ, the walk from s following the labels $\text{PRG}(U_r)$ visits T with probability at least $(\delta\gamma)^a/(ND)^\alpha$. Then $\mathbf{RL} = \mathbf{L}$ and $\mathbf{searchRL} = \mathbf{searchL}$.*

Note that the above theorem requires that the length ℓ of the pseudorandom walks have limited dependence on N and D, being bounded by $(ND)^\alpha$ rather than being polynomial or even linear in ND. Still, this is a much milder requirement than what is achieved by our generator for consistently labelled graphs (Thm. 6.3), which achieves logarithmic dependence. We also note that a pseudorandom generator for logspace algorithms with logarithmic seed length would imply the above theorem, because a truly random walk of length $O(1/\gamma) \cdot O(\log(ND/\delta))$ would end at T with probability at least $\delta/2$, and such a walk can be implemented in space $O(\log(ND/\delta\gamma))$.

Roughly speaking, we will prove Theorem 7.1 by showing that for every poly-mixing graph G, there exists a regular digraph G_{reg} such that the correctness of the generator on G_{reg} implies the correctness of (a modification of) the generator on G. Thus, if we have a generator that works well on regular digraphs, we obtain a generator that works well on instances of POLY-MIXING S-T CONNECTIVITY, which we have shown to be **RL**-complete (Theorem 3.1). The construction of G_{reg} from G is given by the following lemma. We stress that this construction is only done in the *analysis*, and thus need not be computable in log-space.

LEMMA 7.2. *There is a universal constant c such that the following holds. Let $G = (V, E)$ be any d-outregular graph on n vertices with vertices $s, t \in V$ and stationary distribution π such that $\pi(s) \geq 1/k$, $\pi(t) \geq 1/k$, and $\lambda_\pi(G) \leq 1 - 1/k$. Then for every $\varepsilon > 0$, if we set $N_{\text{reg}} = (ndk/\varepsilon)^c$, $D_{\text{reg}} = c \cdot N_{\text{reg}}/\varepsilon$, $\gamma = 1/(ndk)^c$, there is a $(N_{\text{reg}}, d \cdot D_{\text{reg}}, 1-\gamma)$-regular digraph G_{reg} such that the following holds. The vertex set of G_{reg} can be decomposed into "clouds" $V_{\text{reg}} = \bigcup_{v \in V} C_v$ with $|C_s|, |C_t| \geq |V_{\text{reg}}|/2k$. There is a bad set of edge labels $B \subseteq [d] \times [D_{\text{reg}}]$ of density ε such that for every $u \in V$, vertex $\hat{u} \in C_u$ and edge label $(i, j) \in ([d] \times [D_{\text{reg}}]) \setminus B$, the (i, j)'th neighbor of $\hat{u}$ in G_{reg} is in cloud C_v where v is the i'th neighbor of u in G_{reg}.*

The proof of this lemma can be found in the full version [21]; here we show how it implies Theorem 7.1.

Proof of Theorem 7.1: Let $(G, s, t, 1^k)$ be any instance of POLY-MIXING FIND PATH, where G is d-outregular, has n vertices, and has (promised) stationary distribution π with $\pi(s), \pi(t) \geq 1/k$ and $\lambda_\pi(G) \leq 1 - 1/k$. Set $\delta = 1/2k$, and $\varepsilon = 1/(ndk)^b$ for a large constant b to be specified later, and let $N_{\text{reg}} = (ndk/\varepsilon)^c$, $D_{\text{reg}} = c \cdot N_{\text{reg}}/\varepsilon$, $\gamma = 1/(ndk)^c$ be the parameters of the regular digraph guaranteed by Lemma 7.2. Let $\text{PRG} = \text{PRG}_{N_{\text{reg}}, d \cdot D_{\text{reg}}, \delta, \gamma} : \{0,1\}^r \to ([d] \times [D_{\text{reg}}])^\ell$ be the generator hypothesized in Theorem 7.1, with seed length $r = a\log(N_{\text{reg}} D_{\text{reg}}/\delta\gamma) = O(abc\log(ndk))$. and walk length $\ell = (1/\gamma\delta)^a \cdot (N_{\text{reg}} \cdot dD_{\text{reg}})^\alpha = (ndk)^{O(ac)} \cdot (ndk/\varepsilon)^{O(\alpha c)} = (ndk)^{O(ac)}/\varepsilon^{O(\alpha c)}$. Without loss of generality we may assume that each component in $\text{PRG}(U_r)$ is uniformly distributed in $[d] \times [D_{\text{reg}}]$. (Shift each component of the output by adding a random element $s \leftarrow [d] \times [D_{\text{reg}}]$. This only increases the seed length by a constant factor and preserves the pseudorandomness of the output because it is equivalent to shifting all labels in the regular digraph by $-s$.)

The algorithm for POLY-MIXING FIND PATH works as follows. We enumerate the $2^r = (nkd)^{O(abc)}$ seeds of PRG,

for each obtaining a walk $\hat{w} \in ([d] \times [D_{\rm reg}])^\ell$ of length $\ell = (nkd)^{O(abc)}$. Taking the first components of each step in $\hat{w}$, we obtain an induced walk $w \in [d]^\ell$, which we perform on G, starting at s. If any of these walks end at t, we output that walk.

To analyze this algorithm, we consider a walk $\hat{w} \leftarrow \mathrm{PRG}(U_r)$ taken in $G_{\rm reg}$, starting at any vertex of C_s. Since $\lambda(G_{\rm reg}) \leq 1-\gamma$, C_t has density at least $1/2k$, and $\delta = 1/2k$, such a walk will end in C_t with probability at least

$$(1/\delta\gamma)^a \cdot (N_{\rm reg} \cdot dD_{\rm reg})^\alpha = \varepsilon^{O(\alpha c)}/(ndk)^{O(ac)}.$$

We now argue that the induced walk w in G will end at t with nearly the same probability. By the properties of $G_{\rm reg}$, this will be the case provided the walk $\hat{w}$ does not use any edge label from B. Since B has density at most ε and each edge label in $\hat{w}$ is uniformly distributed, the probability any label from B is used is at most

$$\ell \cdot \varepsilon = (ndk)^{O(ac)} \cdot \varepsilon^{1-O(\alpha c)}.$$

Thus the walk w in G ends at t with probability at least

$$\frac{\varepsilon^{O(\alpha c)}}{(ndk)^{O(ac)}} - (ndk)^{O(ac)} \cdot \varepsilon^{1-O(\alpha c)} > 0,$$

provided $\alpha \leq c/\kappa$ and $\varepsilon \leq (1/ndk)^b$ for a $b > \kappa ac$, where κ is a sufficiently large universal constant. In particular, there exists a seed of PRG that will produce a walk from s to t. ■

Acknowledgments

We are grateful to Irit Dinur for her invaluable collaboration during the early stages of this work. We also thank David Zuckerman, Eyal Rozenman, David Karger, and Nati Linial for helpful discussions and suggestions.

8. REFERENCES

[1] M. Ajtai, J. Komlós, and E. Szemerédi. Deterministic simulation in LOGSPACE. In *Proc. 19th STOC*, pp. 132–140, 1987.

[2] R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovász, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *Proc. 20th FOCS*, pp. 218–223, 1979.

[3] R. Armoni, A. Ta-Shma, A. Wigderson, and S. Zhou. An $o(log(n)^{4/3})$ space algorithm for (s,t) connectivity in undirected graphs. *J. ACM*, 47(2):294–311, 2000.

[4] L. Babai, N. Nisan, and M. Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *J. Computer and System Sci.*, pp. 204–232, 1989.

[5] S. Caracciolo, A. Pelissetto, and A. Sokal. Two remarks on simulated tempering. Unpublished manuscript (see [13]), 1992.

[6] J. A. Fill. Eigenvalue bounds on convergence to stationarity for nonreversible markov chains with an application to the exclusion process. *Ann. Applied Probability*, 1:62–87, 1991.

[7] I. Haitner, D. Harnik, and O. Reingold. On the power of the randomized iterate. *ECCC* TR05-135, 2005.

[8] A. Healy, S. Vadhan, and E. Viola. Using nondeterminism to amplify hardness. In *36th STOC*, pp. 192–201, 2004.

[9] P. Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. In *41st FOCS*, pp. 189–197, 2000.

[10] E. Kaplan, M. Naor, and O. Reingold. Derandomized constructions of k-wise (almost) independent permutations. In *Proc. 8th RANDOM*, Springer LNCS 3624, pp. 354 – 365, 2005.

[11] H. R. Lewis and C. H. Papadimitriou. Symmetric space-bounded computation. *Theor. Comput. Sci.*, 19:161–187, 1982.

[12] L. Lovász. *Combinatorial problems and exercises.* North-Holland, Amsterdam, 2nd ed., 1993.

[13] N. Madras and D. Randall. Markov chain decomposition for convergence rate analysis. *Ann. of Applied Probability*, 12:581–606, 2002.

[14] R. A. Martin and D. Randall. Sampling adsorbing staircase walks using a new markov chain decomposition method. In *Proc. 41st FOCS*, pp. 492–502, 2000.

[15] M. Mihail. Conductance and convergence of markov chains: a combinatorial treatment of expanders. In *Proc. 37th FOCS*, pp. 526–531, 1989.

[16] Nisan. RL ⊆ SC. In *Proc. 24th STOC*, pp. 619–623, 1992.

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

[18] N. Nisan, E. Szemeredi, and A. Wigderson. Undirected connectivity in $o(log^{1.5}n)$ space. In *Proc. 30th FOCS*, pp. 24–29, 1989.

[19] N. Nisan and D. Zuckerman. Randomness is linear in space. *J. Computer and System Sci.*, 52(1):43–52, 1996.

[20] O. Reingold. Undirected st-connectivity in log-space. In *Proc. of the 37th STOC*, pages 376–385, 2005.

[21] O. Reingold, L. Trevisan, and S. Vadhan. Pseudorandom walks in biregular graphs and the RL vs. L problem. *ECCC* TR05-022, 2005.

[22] O. Reingold, S. Vadhan, and A. Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders. *Ann. Mathematics*, 155(1), 2001.

[23] E. Rozenman and S. Vadhan. Derandomized squaring of graphs. In *Proc. 8th RANDOM*, Springer LNCS 3624, pp. 436–447, 2005. See also full version, ECCC TR05-92.

[24] M. Saks. Randomization and derandomization in space-bounded computation. In *IEEE 11th Annual Conference on Structure in Complexity Theory*, 1996.

[25] M. Saks and S. Zhou. $\mathrm{bp_hspace}(S) \subseteq \mathrm{dspace}(S^{3/2})$. *J. Computer and System Sci.*, 58(2):376–403, 1999.

[26] J. Savitch. Relationship between nondeterministic and deterministic tape complexities. *J. Computer and System Sci.*, 4(2):177–192, 1970.

[27] A. Sinclair and M. Jerrum. Approximate counting, uniform generation and rapidly mixing Markov chains. *Inform. and Comput.*, 82(1):93–133, 1989.

[28] D. Sivakumar. Algorithmic derandomization via complexity theory. In *Proc. 34th STOC*, pp. 619–626, 2002.

[29] A. Wigderson. The complexity of graph connectivity. In *Proc. 17th FOCS*, pp. 112–132, 1992.