Derandomized Squaring of Graphs

Eyal Rozenman[1] and Salil Vadhan[2,⋆]

[1] Division of Engineering & Applied Sciences
Harvard University, Cambridge, Massachusetts
[2] Department of Computer Science & Applied Mathematics
Weizmann Institute, Rehovot, Israel

Abstract. We introduce a "derandomized" analogue of graph squaring. This operation increases the connectivity of the graph (as measured by the second eigenvalue) almost as well as squaring the graph does, yet only increases the degree of the graph by a constant factor, instead of squaring the degree.
One application of this product is an alternative proof of Reingold's recent breakthrough result that S-T Connectivity in Undirected Graphs can be solved in deterministic logspace.

1 Introduction

"Pseudorandom" variants of graph operations have proved to be useful in a variety of settings. Alon, Feige, Wigderson, and Zuckerman [AFWZ] introduced "derandomized graph products" to give a more illuminating deterministic reduction from approximating clique to within relatively small (eg constant) factors to approximating clique to within relatively large (eg n^ϵ) factors. Reingold, Vadhan, and Wigderson [RVW] introduced the "zig-zag graph product" to give a new construction of constant-degree expander graphs. The zig-zag product found many applications, the most recent and most dramatic of which is Reingold's deterministic logspace algorithm [Rei] for connectivity in undirected graphs.

In this paper, we present a pseudorandom analogue of graph squaring. The *square* X^2 of a graph X is the graph on the same vertex set whose edges are paths of length 2 in the original graph. This operation improves many connectivity properties of the graph, such as the diameter and mixing time of random walks of the graph (both of which roughly halve). However, the degree of the graph squares. In terms of random walks on the graph, this means that although half as many steps are needed to mix, each step costs twice as many random bits. Thus, there is no savings in the amount of randomness needed for mixing.

Our derandomized graph squaring only increases the degree by a constant factor rather than squaring it. Nevertheless, it improves the connectivity almost as much as the standard squaring operation. The measure of connectivity for which we prove this is the second eigenvalue of the graph, which is well-known to be a good measure of the mixing time of random walks, as well as of graph

⋆ Supported by NSF grant CCF-0133096 and ONR grant N00014-04-1-0478.

expansion. The standard squaring operation squares the second eigenvalue; we prove that the derandomized squaring does nearly as well.

The main new application of our derandomized squaring is a new logspace algorithm for connectivity in undirected graphs, thereby giving a new proof of Reingold's theorem [Rei]. Our algorithm, while closely related to Reingold's algorithm, is arguably more natural. In particular, it can be viewed as applying a natural pseudorandom generator, namely that of Impagliazzo, Nisan, and Wigderson [INW], to random walks on the input graph. This makes the analysis of the space requirements of the algorithm simpler. Reingold's algorithm is based on the *zig-zag product*, and constructs a sequence of graphs with an increasing number of vertices. Our analysis, based on derandomized squaring, only works on the vertex set of the original input graph.

Below we describe the derandomized squaring and its application to undirected s-t connectivity in more detail.

1.1 Derandomized Graph Squaring

Let X be an undirected regular graph of degree K.[1] The square X^2 of X has an edge for every path of length 2 in X. One way to visualize it is that for every vertex v in X, we place a clique on its K neighbours (this connects every pair of vertices that has a length 2 path through v). The degree thus becomes K^2. (Throughout the paper, we allow multiple edges and self-loops.)

In derandomized squaring, we use an auxiliary graph G on K vertices and place it instead of a clique on the K neighbours of every vertex v (thus connecting only *some* of the pairs of vertices which have a length 2 path through v). We denote the resulting graph by $X \circledS G$.

If the degree of G is D, the derandomized square will have degree KD, which will be smaller than K^2. We will see, however, that if G is an expander, then even if D is much smaller than K, the derandomized square of X with respect to G improves connectivity similarly to standard squaring.

Our measure of connectivity is the second eigenvalue $\lambda \in [0,1]$ of (the random walk on) the graph; small λ indicates that the random walk mixes rapidly and that the graph has good expansion (i.e. is highly connected). If the second eigenvalue of X is λ then the second eigenvalue of X^2 is λ^2. The second eigenvalue of the derandomized product is not very far. For example, we prove that it is at most $\lambda^2 + 2\mu$ where μ is the second eigenvalue of G.

1.2 A New Logspace Algorithm for Undirected Connectivity

Recall that the problem of undirected st-connectivity is: given an undirected graph G and two vertices s, t, decide whether there is a path from s to t in G. While the time complexity of this problem is well-understood, the space

[1] Actually, following [RTV], we actually work with regular *directed* graphs in the technical sections of the paper, but thinking of undirected graphs suffices for the informal discussion here.

complexity was harder to tackle. A long line of research, starting with Savitch's deterministic $\log^2$-space algorithm [Sav], and the log-space randomized algorithm [AKL$^+$], culminated in Reingold's optimal deterministic log-space algorithm [Rei] (see his paper and its references for more on the history and applications of this problem). We now shortly describe Reingold's algorithm, then present our algorithm and compare the two.

Reingold's Algorithm. Notice that undirected connectivity is solvable in log-space on bounded-degree graphs with *logarithmic diameter* (simply enumerate all paths of logarithmic length in the graph out of the origin vertex). Examples of graphs with logarithmic diameter are expander graphs, i.e. graphs whose second eigenvalue is bounded away from 1. Reingold's idea is to transform the input graph into a bounded-degree expander by gradually decreasing its second eigenvalue.

A natural attempt would be to square the graph. This indeed decreases the second eigenvalue, but increases the degree. To decrease the degree, Reingold uses the *zig-zag graph product* of [RVW], or the related *replacement product*. We describe his algorithm in terms of the latter product.

Given a K-regular graph X on N vertices, and an auxiliary D-regular graph G on K vertices, the replacement product $X \circledR G$ is a $D+1$-regular graph on NK vertices. On each edge (v,w) in X put two vertices, one called e_v "near" v and another called e_w "near" w, for a total of NK vertices. This can be thought of as splitting each vertex v into K vertices forming a "cloud" near v. Place the graph G on each cloud. Now for each edge $e=(v,w)$ of X, put an edge between e_v and e_w. The result is a $(D+1)$-regular graph. Notice that $X \circledR G$ is connected if and only if both X and G are.

The replacement product reduces the degree from K to $D+1$. It is proven in [RVW] (and also follows from [MR]) that when G is a good enough expander, replacement product roughly preserves the second eigenvalue of X. Suppose that X is $(D+1)$ regular and G has $(D+1)^2$ vertices and degree D. Then $X^2 \circledR G$ is again a $(D+1)$-regular graph, whose second eigenvalue is roughly the square of the second eigenvalue of X . Iterating this procedure $\log N$ times leads to a constant-degree expander on *polynomially many vertices*, since at each iteration the number of vertices grows by a factor of about D^2. On the resulting expander we can therefore solve connectivity in logarithmic space. (One also must confirm that the iterations can be computed in logarithmic space as well).

Our Algorithm. Our approach also follows from this idea of increasing connectivity by squaring the graph. However, instead of squaring, and then reducing the degree by a zigzag product (and thus increasing the number of vertices) we will replace the squaring by derandomized squaring, which maintains the vertex set (but increases the degree). Iterating the derandomized squaring operation yields highly connected graphs with relatively small degree compared to doing the same iterations with standard squaring. In the next two paragraphs we compare the resulting graphs in each case.

Let X be a regular graph on N vertices. Squaring the graph $\log N$ times, results in the graph $X^{2^{\log N}} = X^N$ (whose edges are all paths of length N in X). This graph is extremely well connected; it contains an edge between every two vertices which are connected by a path in X. The degree however, is huge — exponential in N. We want to simulate the behavior of X^N with a graph that has much smaller degree.

Suppose that instead of standard squaring at each step we apply derandomized squaring to obtain a sequence of graphs $X_1, X_2, \ldots$. At each step the degree increases by a constant factor (instead of the degree squaring at each step). For $m = O(\log N)$ the degree of X_m is only *polynomial* in N. But we will show that is as well-connected as X^N (as measured by the second eigenvalue). In particular, X_m will contain an edge between every pair of vertices s, t that are connected by a path in X. Deciding whether s, t are connected therefore reduces to enumerating all neighbors of s in X_m and looking for t. There are only polynomially many neighbors, so the search can be done in logarithmic space. We will show that computing neighbors in X_m can also be done in logarithmic space. These two facts yield a logarithmic space algorithm for undirected connectivity.

Comparing our approach to Reingold's original solution, the main way in which our algorithm differs from (and is arguably more natural than) Reingold's algorithm is that all the graphs we construct are on the *same* vertex set. Edges in the graph X_m correspond to paths of length 2^m in X. The price we pay is that the degree increases, but, thanks to the use of *derandomized* squaring, only by a constant factor (which we can afford). In contrast, each step of Reingold's algorithm creates a graph that is larger than the original graph (but maintains constant degree throughout).

1.3 Derandomized Squaring as a Pseudo-random Generator

Impagliazzo, Nisan, and Wigderson [INW] proposed the following pseudorandom generator. Let G be an expander graph with K vertices and degree D. Choose a random vertex $x \leftarrow [K]$, a random edge label $a \leftarrow [D]$, and output $(x, x[a]) \in [K] \times [K]$. This pseudorandom generator is at the heart of derandomized squaring. Notice that using this pseudorandom generator to generate a pseudorandom walk of length 2 in a graph X of degree K is *equivalent* to taking a random step in the derandomized square of X using auxiliary graph G.

Impagliazzo, Nisan, and Wigderson [INW] suggested to increase the stretch of the above generator by recursion. They proved that when the graphs G used in the construction are sufficiently good expanders of relatively *large degree*, this construction fools various models of computation (including randomized logspace algorithms)[2]. However, the resulting generator has seed length $O(\log^2 n)$, and hence does not prove that RL=L.

Our construction of the graph X_m in our st-connectivity algorithm is precisely the graph corresponding to using the INW generator to derandomize random

[2] Specifically, to fool an algorithm running in space $\log n$, they use expanders of degree $\mathrm{poly}(n)$.

walks of length 2^m in X. However, we are able to use *constant-degree* expanders for G (for most levels of recursion), thereby obtaining seed length $O(\log n)$ and hence a logspace algorithm (albeit for undirected st-connectivity rather than all of RL).

Moreover, it follows from our analysis that taking the pseudorandom walk in X corresponding to a random step in X_m (equivalently, according to the INW generator with appropriate parameters) will end at an almost-uniformly distributed vertex. A pseudorandom generator with such a property was already given in [RTV] based on Reingold's algorithm and the zig-zag product, but again it is more natural in terms of derandomized squaring.

1.4 Relation to the Zig-Zag Product

The reader may have noticed a similarity between the derandomized squaring and the zig-zag product of [RVW] (which we define precisely later in the paper). Indeed, they are very closely related. When we use a square graph G^2 as auxiliary graph, the derandomized square $X \circledS G^2$ turns out to be a "projection" of the square of the zigzag product $(X \text{ⓩ} G)^2$. This observation allows us to prove the expansion properties of the derandomized squaring by reducing to the zig-zag product case.

We note that the derandomized squaring has complementary properties to the zigzag product. In the zigzag product we are given a graph X and can decrease its degree while (nearly) maintaining its second eigenvalue. We must pay by slightly increasing the number of vertices. In the derandomized squaring we manage to decrease the second eigenvalue while maintaining the number of vertices, and we pay by slightly increasing the degree.

2 Preliminaries

Reingold, Trevisan, and Vadhan [RTV] generalized Reingold's algorithm and the zig-zag product to (regular) *directed* graphs, and working in this more general setting turns out to be useful for us, too (even if we are only interested in solving st-connectivity for undirected graphs). We present the necessary background on such graphs in this section.

Let X be a directed graph (*digraph* for short) on N vertices. We say that X is K*-outregular* if every node has outdegree K, K*-inregular* if every node has indegree K, and K*-regular* if both conditions hold. Graphs may have self-loops and multiple edges, where a self-loop is counted as both an outgoing and incoming edge. All graphs in this paper are outregular directed graphs (and most are regular).

For a K-regular graph X on N vertices, we denote by M_X the transition matrix of the random walk on X, i.e. the adjacency matrix divided by K. Let $u = (1/N, \ldots, 1/N) \in \mathbb{R}^N$ be the uniform distribution on the vertices of X. Then, by regularity, $M_X u = u$ (so u is an eigenvector of eigenvalue 1).

Following [Mih], we consider the following measure of the rate at which the random walk on X converges to the stationary distribution u:

$$\lambda(X) = \max_{v \perp u} \frac{\|M_X(v)\|}{\|v\|} \in [0,1]$$

where $v \perp u$ refers to orthogonality with respect to the standard dot product $\langle x, y\rangle = \sum_i x_i y_i$ on $\mathbb{R}^N$ and $\|x\| = \sqrt{\langle x, x\rangle}$ is the L_2 norm. The smaller $\lambda(X)$, the faster the random walk converges to the stationary distribution and the better "expansion" properties X has. Hence, families of graphs with $\lambda(X) \leq 1 - \Omega(1)$ are referred to as *expanders*.

In case X is undirected, $\lambda(X)$ equals the second largest eigenvalue of the symmetric matrix M_X in absolute value. In directed graphs, it equals the square root of the second largest eigenvalue of $M_X^T M_X$.

A K-regular directed graph X on N vertices with $\lambda(X) \leq \lambda$ will be called an (N, K, λ)-graph. We define $\mathrm{g}(X) = 1 - \lambda(X)$ to be the *spectral gap* of X.

A *labelling* of a K-outregular graph X is an assignment of a number in $[K]$ to every edge of X, such that the edges exiting every vertex have K distinct labels. For a vertex v of X and an edge label $x \in [K]$ we denote by $v[x]$ the neighbor of v via the outgoing edge labelled x. We say that a labelling is *consistent* if for every vertex all incoming edges have distinct labels. Notice that if a graph has a consistent labeling, then it is K-inregular (and hence K-regular). We will work with consistently labelled graphs in this extended abstract for simplicity and to make the connection between derandomized squaring and the INW pseudorandom generator [INW] more apparent. But this condition condition can be relaxed (eg in the definition and analysis of the derandomized square) by allowing each edge (u, v) to have two labels, one as an outgoing edge from u and one as an incoming edge to v, as formalized using the "rotation maps" of [RVW, RTV].

The notion of consistent labelling we use is the same as in [HW] and [RTV]. For undirected graphs, one can consider a stronger notion of consistent labelling, where an edge (u, v) is required to have the same label leaving from u as it does when leaving v, but this has the disadvantage that it is not preserved under the operations we perform (such as the squaring operation below).

The square X^2 of a graph X is the graph whose edges are paths of length 2 in X. The square of a K-regular graph is K^2-regular, and a consistent labelling of X induces a consistent labelling of X^2 in a natural way. Specifically, for a label $(x, y) \in [K]^2$, we define $v[x, y] = v[x][y]$. Notice that $\lambda(X^2) \leq \lambda(X)^2$. (This is always an equality for undirected graphs, but not necessarily so for directed graphs). We similarly define the n-th power X^n using paths of length n in X.

Like undirected graphs, the spectral gap of *regular* connected directed graphs is always at least an inverse polynomial. This holds provided the graph is connected and aperiodic (i.e. the gcd of all cycle-lengths is 1). (If either of these conditions doesn't hold than $\lambda(X) = 1$.) The following can be proven by reduction to the corresponding lemma in the undirected case [AS].

Lemma 2.1. *For every K-regular, connected, aperiodic graph. Then $\lambda(X) \leq 1 - 1/(2D^2N^2)$.*

The next proposition shows that when the second eigenvalue is very small, the graph is very well connected - it contains a clique (we omit the proof).

Proposition 2.2. *An $(N, D, 1/2N^{1.5})$-graph X contains an edge between any pair of vertices. Indeed, for a pair of vertices v, w the probability that a random neighbor of v is equal w is at least $1/N - 1/N^2$.*

3 Derandomized Squaring

After giving a formal definition of derandomized squaring, we will show that it decreases the second eigenvalue of a graph in a way comparable to squaring it. The proof is by reduction to the zigzag product $X \circled{z} G$ (also defined below), of X with an auxiliary graph G. We shall see that the derandomized square of X with the auxiliary squared graph G^2 is a "projection" (defined precisely later) of the squared graph $(X \circled{z} G)^2$. We then show that projection does not increase the second eigenvalue. We can therefore use the known bounds on the second eigenvalue from [RTV].

Definition 3.1. *Let X be a labelled K-regular graph on vertex set $[N]$, let G be a labelled D-regular graph on vertex set $[K]$. The* derandomized square graph *$X \circled{s} G$ has vertex set $[N]$ and is KD-outregular. The edges exiting a vertex v are paths $v[x][y]$ of length two in X such that y is a neighbor of x in G. Equivalently, when $x \in [K]$ is an edge label in X and $a \in [D]$ is an edge label in G, the neighbor of $v \in [N]$ via the edge labelled (x, a) is $v[x][x[a]]$.*

The derandomized square may, in general, not produce an in-regular graph. However, it will do so provided that X is consistently labelled.

Proposition 3.2. *If X is consistently labelled, then $X \circled{s} G$ is KD-regular. If, in addition, G is consistently labelled, then $X \circled{s} G$ is consistently labelled.* ∎

Notice that even if X and G are consistently labelled and undirected, i.e. for every edge (u, v) there is a corresponding reverse edge (v, u), then the derandomized square $X \circled{s} G$ need not be undirected[3].

Our main result is that when G is a good expander, then the expansion of $X \circled{s} G$ is close to that of X^2.

Theorem 3.3. *If X is an (N, K, λ)-graph and G is a (K, D, μ)-graph, then $X \circled{s} G^2$ is an $(N, KD^2, f(\lambda, \mu))$-graph, for a function f, monotone increasing in λ and μ, and satisfying*

- $f(\lambda, \mu) \leq \lambda^2 + 2\mu^2$,
- $f(1 - \gamma, 1/100) \leq 1 - (8/7) \cdot \gamma$, *when* $\gamma < 1/4$.

We will prove the theorem via reduction to the zigzag product $X \circled{z} G$ which we now turn to define[4].

[3] If X satisfied the stronger notion of consistent labeling where (u, v) and (v, u) are required to have the same label, then $X \circled{s} G$ would be undirected. Alas, this stronger notion is not preserved under the derandomized square (or even the standard squaring).

[4] In the full version, we will provide a direct proof, which proceeds along similar lines to the zig-zag analysis of [RVW, RTV], but is simpler and provides a better bound.

Definition 3.4 ([RVW]). *Let X be an labelled K-regular graph on vertex set $[N]$. Let G be a labelled D-regular graph on vertex set $[K]$. The* zig-zag product *$X \text{ⓩ} G$ is a graph on vertex set $[N] \times [K]$ of outdegree D^2, where the edge label $(a,b) \in [D] \times [D]$ connects vertex $(v,x) \in [N] \times [K]$ to $(v[x[a]], x[a][b])$.*

Again, consistent labelling of X ensures that $X \text{ⓩ} G$ is a regular graph. We will prove the following lemma.

Lemma 3.5. *Let X be consistently labelled. Then $\lambda(X \text{ⓢ} G^2) \leq \lambda(X \text{ⓩ} G)^2$.*

Theorem 3.3 then follows by plugging in the bounds on $\lambda(X \text{ⓩ} G)$ given by [RVW, RTV].
Proof of Lemma 3.5: We will show that $X \text{ⓢ} G^2$ is a projection (defined next) of $(X \text{ⓩ} G)^2$, and prove that projection cannot increase the second eigenvalue.

Definition 3.6. *Let X be a D-regular graph on vertex set $[N] \times [K]$. The* **projection graph** *$\mathrm{P}X$ has vertex set $[N]$ and edge labels $[K] \times [D]$. The neighbor labelled (x,a) of a vertex v is defined by $v[x,a] = (v,x)[a]_1$, where the right-hand side refers to the first component of the a'th neighbor of (v,x) in X. The projection $\mathrm{P}X$ is a KD-regular graph, but the labeling above is not necessarily consistent.*

Proposition 3.7. *The projection graph $\mathrm{P}(X \text{ⓩ} G)^2$ is equal to the graph obtained from $X \text{ⓢ} G^2$ by duplicating each edge K^2 times.*

We omit the proof, which is a straightforward verification from the definitions.

Proposition 3.8. $\lambda(\mathrm{P}X) \leq \lambda(X)$.

Proof. Let $f : [N] \to \mathbb{R}$ be a vector of $\mathrm{P}X$ that is orthogonal to the uniform distribution (i.e. $\sum_i f(i) = 0$) and $\|A_{\mathrm{P}X} f\| = \lambda \cdot \|f\|$, where $\lambda = \lambda(\mathrm{P}X)$.

Define $\hat{f} : [N] \times [K] \to \mathbb{R}$ by $\hat{f}(v,x) = f(v)$. Observe that $f(v[x,a]) = \hat{f}(v,x)[a]$. So

$$\frac{1}{K} \sum_{x \in [K]} M_X \hat{f}(v,x) = \frac{1}{KD} \sum_{\substack{a \in [D] \\ x \in [K]}} \hat{f}((v,x)[a]) = \frac{1}{KD} \sum_{\substack{a \in [D] \\ x \in [K]}} f(v[x,a]) = M_{\mathrm{P}X} f(v).$$

The leftmost expression above is a vector on $[N] \times [K]$. It is obtained by applying an average operator to the vector $M_X \hat{f}$, which cannot increase the L_2 norm. Therefore $\|M_X \hat{f}\|$ is at least $\lambda \|\hat{f}\|$. The sum of coordinates of $\hat{f}$ is zero, so $\lambda(X) \geq \lambda$. ∎

4 A Log-Space Algorithm for Undirected Connectivity

We describe how to solve undirected st-connectivity on an undirected graph X with N vertices in logarithmic space.

Overview. We will assume that the input graph X is 3-regular and consistently labelled. (in the full version we will show that this assumption does not lose generality). By Lemma 2.1, every 3-regular connected graph has second eigenvalue $1 - \Omega(1/N)$. Our goal is to use derandomized squaring to decrease the second eigenvalue (of each connected component) to less than $1/2N^3$ (we will need to square $O(\log N)$ times). By Prop. 2.2, the resulting graph must contain a clique on every connected component of X. We can therefore go over all the neighbors of s in the resulting graph and look for t.

Starting with (some power of) X, we define a sequence of graphs X_m, each of which is a derandomized square of its predecessor using a suitable auxiliary graph. The algorithm works in two phases. Phase one works for $m \leq 100 \log N$, and reduces the second eigenvalue to a constant (3/4), by using as auxiliary graphs a sequence G_m of *fixed-degree* expanders. We will see that the spectral gap $\mathrm{g}(X_m)$ grows by at least a factor of 8/7 at each step. Therefore, after $m_0 = O(\log N)$ steps, we obtain an expander X_{m_0} with second eigenvalue at most 3/4 and degree polynomial in N.

At this point we cannot use fixed-degree expanders as auxiliary graphs any more. If we did, the second eigenvalue of the derandomized square would be dominated by the second eigenvalue of the auxiliary graph, which is constant. Thus we would not be able to decrease the eigenvalue to $1/2N^3$. In phase two, we therefore use auxiliary graphs G_m with non-constant degrees. Specifically, for $m > m_0$, the auxiliary graph G_m will have degree doubly-exponential in $m - m_0$. The fast growth of the degree allows the eigenvalue of the auxiliary graph to remain small enough to imply that $\lambda(X_{m+1}) \leq c \cdot \lambda(X_m)^2$ for some $c > 1$ quite close to 1. Therefore, after an additional $\log\log N + O(1)$ steps we obtain a graph X_{m_1} with second eigenvalue at most $1/2N^3$.

Since the graph X_{m_1} has degree polynomial in N, we can enumerate all the neighbors of s in logarithmic space. We will show (in Prop. 4.4) that neighbors in X_{m_1} are log-space computable, making the whole algorithm work in logarithmic space.

The Auxiliary Expanders. We will need a family of logspace-constructible constant-degree expanders with the following parameters, (which can be obtained from e.g. [GG] or [RVW].

Lemma 4.1. *For some constant $Q = 3^q$, there exists a sequence H_m of consistently labeled $(Q^{2m}, Q, 1/100)$-graphs. Neighbors in H_m are computable in space $O(m)$ (i.e. given a vertex name $v \in [Q^{2m}]$ and an edge label $x \in [Q]$, we can compute $v[x]$ in space $O(m)$).*

Definition 4.2. *Let H_m be the graph sequence of lemma 4.1. For a positive integer N, we set $m_0 = \lceil 100 \log N \rceil$, we define a graph sequence G_m by $G_m = (H_m)^2$ for $m \leq m_0$, and $G_m = (H_{m_0+2^{m-m_0-1}})^{2^{m-m_0}}$ for $m > m_0$. Neighbors in G_m are computable in space $O(m + 2^{m-m_0})$.*

The Algorithm. Let X be a 3-regular consistently labelled graph. Given two vertices s, t connected in X, we describe a log-space algorithm that outputs a path between s and t. For simplicity, assume that X is connected (else carry out the analysis below on each connected component of X).

Define $X_1 = X^{2q}$, where $Q = 3^q$ is from lemma 4.1. Define inductively $X_{m+1} = X_m \circledS G_m$. It can be verified by induction that the degree D_m of X_m is equal to the number of vertices of G_m, so the operation $X_m \circledS G_m$ is indeed well-defined. Specifically, we have $D_m = Q^{2m}$ for $m \leq m_0$, and $D_m = Q^{2\cdot(m_0+2^{m-m_0})}$ for $m > m_0$.

Phase One. By lemma 2.1 we have $g(X_1) \geq 1/3N$. From the second inequality in theorem 3.3 $\mathrm{g}(X_m) \geq \mathrm{g}(X_{m-1}) \cdot (8/7)$ as long as $\mathrm{g}(X_{m-1}) \leq 1/4$. Therefore for some $m < 100 \log N$ we will get $g(X_m) \leq 1/4$. Because of the monotonicity mentioned in theorem 3.3 the gap does not decrease in the proceeding iterations. Therefore, for $m_0 = \lceil 100 \log N \rceil$ we have $\lambda(X_{m_0}) < 3/4$.

Phase Two. We now decrease the second eigenvalue from $3/4$ to $1/2N^3$. For $m \geq m_0$ define $\lambda_m = (64/65) \cdot (7/8)^{2^{(m-m_0)}}$, $\mu_m = (1/100)^{2^{m-m_0}}$. Suppose that $\lambda(X_m) \leq \lambda_m$. Since $2\mu_m^2 < \lambda_m^2/64$ we can use the first inequality in theorem 3.3 to deduce that $\lambda(X_{m+1}) \leq \lambda_m^2(1 + 1/64)$. Since for $m = m_0$ indeed $\lambda(X_m) \leq \lambda_m$ we deduce this holds for all $m \geq m_0$. The next proposition is a direct consequence.

Proposition 4.3. *Let $m_1 = m_0 + \log\log N + 10$. Then $\lambda(X_{m_1}) \leq 1/2N^3$.* ∎

By Prop. 2.2 the graph X_{m_1} contains a clique on the N vertices. Moreover, it has degree $D_{m_1} = Q^{2\cdot(100 \log N + 2^{\log\log N+10})} = \mathrm{poly}(N)$. If we could compute neighbors in X_{m_1} in space $O(\log N)$ we could find a path from s to t in logarithmic space.

Proposition 4.4. *Neighborhoods in X_{m_1} are computable in space $O(\log N)$.*

Proof. Edge labels in X_m are vectors $y_m = (y_1, a_1, \ldots, a_{m-1})$ where y_1 is an edge label in X_1 and a_i is an edge label on G_i. Given a vertex v and an edge label y_m in X_m we wish to compute the neighbor $v[y_m]$ in X_m.

Every edge in X_m corresponds to a path of length 2^m in X. It suffices to give a (log-space) algorithm that, given v,y and an integer b in the range $[1, 2^m]$, returns the edge label in X of the b-th edge in this path of length 2^m. As we will see below, this edge label is actually independent of the vertex v (and thus can be computed given only y and b).

The path of length 2^m originating from v corresponding to the edge label y_m consists of two paths of length 2^{m-1} corresponding to two edges in X_{m-1}. These two edges in X_{m-1} have labels $y_{m-1} = (y_1, a_1, \ldots, a_{m-2})$ and $y_{m-1}[a_{m-1}]$, where the latter is a neighbor computation in G_{m-1}.

From these observations the algorithm is simple. If $b \leq 2^{m-1}$ then solve the problem encoded by y_{m-1}, b in X_{m-1}. If $b > 2^{m-1}$ then instead set $y_{m-1} \leftarrow y_{m-1}[a_{m-1}]$, $b \leftarrow b - 2^{m-1}$, and now solve the problem encoded by y_{m-1}, b on X_{m-1}.

Here is a pseudo code for the algorithm. Write $b - 1$ as a binary string $(b_{m-1}, \ldots, b_0)$, and let y_i be the string $y_1, a_1, \ldots, a_{i-1}$.

```
for $i = m - 1$ to $0$ do
  if $b_i = 1$ then
    set $y_i = y_i[a_i]$ (this is a computation in $G_i$).
  end if
end for
output $y_0$
```

Now we argue that this can be computed in space $O(\log N)$ when $m = m_1$. Notice that the input length to the algorithm is $m + \log D_{m_1} = O(\log N)$. By lemma 4.1, the computation in G_i steps in the loop described in the code can be performed in space $O(m + 2^{m-m_0}) = O(\log N)$, and we are done. ∎

A Pseudo-random Generator for Walks on X. Picking a random edge in X_{m_1} yields a walk of length $2^{m_1} = \mathrm{poly}(N)$ in X. This walk is "pseudo-random" - it ends at an almost-uniformly distributed vertex (by prop. 2.2 the probability to reach any vertex of X is at least $1/N - 1/N^2$), but is generated by only $\log D_{m_1} = O(\log N)$ random bits (compare with $\mathrm{poly}(N)$ bits required to generate a standard random walk of this length). Moreover, the edge labels in this walk do not depend on the initial vertex v, but only on the edge label chosen in X_{m_1}. Indeed, the algorithm given above describes how to compute the labels in the walk given the edge label y_{m_1} in X_{m_1}. In fact, the map from y_{m_1} to the sequence of edge labels in the walk is precisely the pseudorandom generator constructed in [INW] from the expanders $G_1, \ldots, G_{m-1}$. This pseudo-random walk generator will have the above property (producing walks that end at almost-uniformly distributed vertices) in any consistently labeled graph (of specified parameters). A pseudorandom walk generator with similar properties was given in [RTV] based on Reingold's algorithm (which uses the zig-zag product). However, the generator does not have as simple a description as above. In particular, computing the b'th label produced by the walk seems to require computing all the previous $b-1$ labels of the walk (taking time up to $2^{m_1} = \mathrm{poly}N$, rather than being computable directly as above (in time $\mathrm{poly}(\log N)$).

Acknowledgments

This work emerged from of our collaborations with Omer Reingold, Luca Trevisan, and Avi Wigderson. We are deeply grateful to them for their insights on this topic and their encouragement in writing this paper.

References

[AKL$^+$] 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 *20th Annual Symposium on Foundations of Computer Science (San Juan, Puerto Rico, 1979)*, pages 218–223. IEEE, New York, 1979.

[AFWZ] N. Alon, U. Feige, A. Wigderson, and D. Zuckerman. Derandomized graph products. *Comput. Complexity*, 5(1):60–75, 1995.

[AS] N. Alon and B. Sudakov. Bipartite subgraphs and the smallest eigenvalue. *Combin. Probab. Comput.*, 9(1):1–12, 2000.

[GG] O. Gabber and Z. Galil. Explicit Constructions of Linear-Sized Superconcentrators. *J. Comput. Syst. Sci.*, 22(3):407–420, June 1981.

[HW] S. Hoory and A. Wigderson. Universal Traversal Sequences for Expander Graphs. *Inf. Process. Lett.*, 46(2):67–69, 1993.

[INW] R. Impagliazzo, N. Nisan, and A. Wigderson. Pseudorandomness for Network Algorithms. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 356–364, Montréal, Québec, Canada, 23–25 May 1994.

[MR] R. A. Martin and D. Randall. Sampling Adsorbing Staircase Walks Using a New Markov Chain Decomposition Method. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 492–502, Redondo Beach, CA, 17–19 Oct. 2000. IEEE.

[Mih] M. Mihail. Conductance and convergence of markov chains: a combinatorial treatment of expanders. In *In Proc. of the 37th Conf. on Foundations of Computer Science*, pages 526–531, 1989.

[RTV] Reingold, Trevisan, and Vadhan. Pseudorandom Walks in Biregular Graphs and the RL vs. L Problem. In *ECCCTR: Electronic Colloquium on Computational Complexity, technical reports*, 2005.

[Rei] O. Reingold. Undirected ST-connectivity in log-space. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 376–385, New York, NY, USA, 2005. ACM Press.

[RTV] O. Reingold, L. Trevisan, and S. Vadhan. Pseudorandom Walks in Biregular Graphs and the RL vs. L Problem. *Electronic Colloquium on Computational Complexity* Technical Report TR05-022, February 2005. `http://www.eccc.uni-trier.de/eccc`.

[RVW] O. Reingold, S. Vadhan, and A. Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders. *Ann. of Math. (2)*, 155(1):157–187, 2002.

[Sav] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. System. Sci.*, 4:177–192, 1970.