THEORY OF COMPUTING, Volume 17 (4), 2021, pp. 1–35
www.theoryofcomputing.org

SPECIAL ISSUE: APPROX-RANDOM 2019

Deterministic Approximation of Random Walks in Small Space

Jack Murtagh* Omer Reingold† Aaron Sidford ‡ Salil Vadhan §

Received February 6, 2020; Revised September 19, 2020; Published September 18, 2021

Abstract. We give a deterministic, nearly logarithmic-space algorithm that given an undirected multigraph G, a positive integer r, and a set S of vertices, approximates the conductance of S in the r-step random walk on G to within a factor of $1+\varepsilon$, where $\varepsilon > 0$ is an arbitrarily small constant. More generally, our algorithm computes an ε-spectral approximation to the normalized Laplacian of the r-step walk.

Our algorithm combines the derandomized square graph operation (Rozenman and Vadhan, RANDOM'05), which we recently used for solving Laplacian systems in nearly logarithmic space (Murtagh et al., FOCS'17), with ideas from (Cheng et al., 2015) which gave an algorithm that is time-efficient (while ours is space-efficient) and randomized (while ours is deterministic) for the case of even r (while ours works for all r). Along the way, we provide some new results that generalize technical machinery and yield improvements over

A conference version of this paper appeared in the Proceedings of the 23rd International Conference on Randomization and Computation (RANDOM 2019) [31].

*Supported by NSF grant CCF-1763299.

†Supported by NSF grant CCF-1763311.

‡Supported by NSF CAREER Award CCF-1844855.

§Supported by NSF grant CCF-1763299.

ACM Classification: F.2.2, G.2.2

AMS Classification: 68Q25, 68R10

Key words and phrases: random walks, space complexity, derandomization, expander graphs, spectral approximation

 DOI: 10.4086/toc.2021.v017a004

previous work. First, we obtain a nearly linear-time randomized algorithm for computing a spectral approximation to the normalized Laplacian for odd r. Second, we define and analyze a generalization of the derandomized square for irregular multigraphs and for sparsifying the product of two distinct multigraphs. As part of this generalization, we also give a strongly explicit construction of expander graphs of every size.

1 Introduction

Random walks provide the most dramatic example of the power of randomized algorithms for solving computational problems in the space-bounded setting, as they only require logarithmic space (to store the current state or vertex). In particular, since undirected graphs have polynomial cover time, random walks give a randomized logspace (**RL**) algorithm for UNDIRECTED S-T CONNECTIVITY [2]. Reingold [34] showed that this algorithm can be derandomized, and hence that UNDIRECTED S-T CONNECTIVITY is in deterministic logspace (**L**). However, Reingold's algorithm does not match the full power of random walks on undirected graphs; in particular it does not allow us to approximate properties of the random walk at lengths below the mixing time.

In this article, we provide a nearly logarithmic-space algorithm for approximating properties of arbitrary-length random walks on an undirected graph, in particular the *conductance* of any set of vertices:

Definition 1.1. Let $G=(V,E)$ be an undirected graph, r a positive integer, and $S \subseteq V$ a set of vertices. The *conductance* of S under the r-step random walk on G is defined as

$$\Phi_r(S) = \Pr[V_r \notin S | V_0 \in S],$$

where $V_0, V_1, \ldots, V_r$ is a random walk on G started at the stationary distribution $\Pr[V_0 = v] = \deg(v)/2|E|$.

Theorem 1.2. *There is a deterministic algorithm that given an undirected multigraph G on n vertices, a positive integer r, a set of vertices S, and $\varepsilon > 0$, computes a number $\widetilde{\Phi}$ such that*

$$(1-\varepsilon)\cdot\Phi_r(S) \le \widetilde{\Phi} \le (1+\varepsilon)\cdot\Phi_r(S)$$

and runs in space $O(\log N + (\log r)\cdot\log(1/\varepsilon) + (\log r)\cdot\log\log r)$, where N is the bit length of the input graph G (the length of the description of G as a list of edges, in a binary encoding).

Previously, approximating conductance could be done in $O(\log^{3/2}(N/\varepsilon) + \log\log r)$ space, which follows from the proof by Saks and Zhou that **RL** is in $\mathbf{L}^{3/2}$ [37].

Two interesting parameter regimes where we improve the Saks-Zhou bound are when $r = 1/\varepsilon = 2^{O(\sqrt{\log N})}$, in which case our algorithm runs in space $O(\log N)$, or when $\varepsilon = 1/\mathrm{polylog}(N)$ and $r \le \mathrm{poly}(N)$, in which case our algorithm runs in space $\widetilde{O}(\log N)$. When r exceeds the $\mathrm{poly}(N)\cdot\log(1/\varepsilon)$ time for random walks on undirected graphs to mix to within distance ε of the stationary distribution, the conductance can be approximated in space $O(\log(N/\varepsilon) + \log\log r)$ by using Reingold's algorithm to find the connected components of G, and the bipartitions of the components that are bipartite and calculating the stationary probability of S restricted to each of these pieces, which is proportional to the sum of degrees of vertices in S.

We prove Theorem 1.2 by providing a stronger result that with the same amount of space it is possible to compute an ε-*spectral approximation* to the *normalized Laplacian* of the r-step random walk on G.

Definition 1.3. Let G be an undirected graph with adjacency matrix A, diagonal degree matrix D, and transition matrix $T = AD^{-1}$. The *transition matrix for the r-step random walk* on G is T^r. The *normalized Laplacian* of the r-step random walk is the symmetric matrix $I - M^r$ for $M = D^{-1/2}AD^{-1/2}$.

Spectral approximation of the normalized Laplacian is a strong definition that guarantees the two matrices have similar eigenvalues, and their corresponding graphs have similar cuts and random walk behavior [39, 9]. Note that the normalized Laplacian can also be expressed as $I - M^r = D^{-1/2}(I - T^r)D^{1/2}$, so it does indeed capture the behavior of r-step random walks on G.[1]

Theorem 1.4 (Main result)**.** *There is a deterministic algorithm that given an undirected multigraph G on n vertices with normalized Laplacian $I - M$, a nonnegative integer r, and $\varepsilon > 0$, constructs an undirected multigraph $\widetilde{G}$ whose normalized Laplacian $\widetilde{L}$ is an ε-spectral approximation of $L = I - M^r$. That is, for all vectors $v \in \mathbb{R}^n$*

$$(1-\varepsilon) \cdot v^T L v \leq v^T \widetilde{L} v \leq (1+\varepsilon) \cdot v^T L v.$$

The algorithm runs in space $O(\log N + (\log r) \cdot \log(1/\varepsilon) + (\log r) \cdot \log\log r)$, where N is the bit length of the input graph G.

Theorem 1.2 follows from Theorem 1.4 by taking v to be $D^{1/2}e_S$ where e_S is the characteristic vector of the set S and normalizing appropriately. (See Section 5).

Our main technique for proving Theorem 1.4 is the *derandomized product*, a new generalization of the *derandomized square*, which was introduced by Rozenman and Vadhan [36] to give an alternative proof that UNDIRECTED S-T CONNECTIVITY is in **L**. Our main result follows from carefully applying the derandomized product and analyzing its properties with inequalities from the theory of spectral approximation. Specifically, our analysis is inspired by the work of Cheng, Cheng, Liu, Peng, and Teng [14], who studied the approximation of random walks by randomized algorithms running in nearly linear time. We emphasize that [14] gives a randomized algorithm with high space complexity (but low time complexity) for approximating properties of even-length walks, while we give a deterministic, space-efficient algorithm for approximating properties of walks of every length. Interestingly, while the graphs in our results are all undirected, some of our analyses use techniques for spectral approximation of *directed* graphs introduced by Cohen, Kelner, Peebles, Peng, Rao, Sidford, and Vladu [16, 15].

The derandomized square can be viewed as applying the pseudorandom generator of Impagliazzo, Nisan, and Wigderson [23] to random walks on labelled graphs. It is somewhat surprising that repeated derandomized squaring does not blow up the error by a factor proportional to the length of the walk being derandomized. For arbitrary branching programs, the INW generator does incur error that is linear in the length of the program. Some special cases such as regular [11, 12, 17] and permutation [17, 40] branching programs of constant width have been shown to have a milder error growth as a function of the walk length. Our work adds to this list by showing that properties of random walks of length k on undirected graphs can be estimated in terms of spectral approximation without error accumulating linearly in k.

In our previous paper [30], we showed that the Laplacian of the derandomized square of a *regular* graph spectrally approximates the Laplacian of the true square, $I - M^2$, and this was used in a recursion from [33] to give a nearly logarithmic-space algorithm for approximately solving Laplacian systems

[1] When G is irregular, the matrix $I - T^r$ is not necessarily symmetric. It is a *directed Laplacian* as defined in [16, 15]. See Definition 2.5.

$Lx = b$. A natural idea to approximate the Laplacian of higher powers, $I - M^r$, is to repeatedly apply the derandomized square. This raises three challenges, and we achieve our result by showing how to overcome each:

1. It is not guaranteed from [30] that repeated derandomized squaring preserves spectral approximation. For this, we use ideas from [14] to argue that it does.

2. When r is not a power of 2, the standard approach would be to write $r = b_0 + 2 \cdot b_1 + \ldots + 2^z \cdot b_z$ where b_i is the i-th bit of r and multiply approximations to M^{2^i} for all i such that $b_i \neq 0$. The problem is that multiplying spectral approximations of matrices does not necessarily yield a spectral approximation of their product. Our solution is to generalize the derandomized square to produce sparse approximations to the product of *distinct* graphs. In particular, given $I - M$ and an approximation $I - \widetilde{M}$ to $I - M^k$, our derandomized product allows us to combine M and $\widetilde{M}$ to approximate $I - M^{k+1}$. Although our generalized graph product is defined for undirected graphs, its analysis uses machinery for spectral approximation of directed graphs, introduced in [15].

3. We cannot assume that our graph is regular without loss of generality. In contrast, [34, 36, 30] could do so, since adding self-loops does not affect connectivity or solutions to Laplacian systems of G, however, it does affect random walks. Our solution is to define and analyze the derandomized product for irregular graphs.

A key element in the derandomized product is a strongly explicit (i. e., neighbor relations can be computed in time $\mathrm{polylog}(N)$ and space $O(\log N)$) construction of expander graphs whose sizes equal the degrees of the vertices in the graphs being multiplied. Many commonly used strongly explicit expander families only contain graph sizes that are certain subsets of $\mathbb{N}$ such as powers of 2 (Cayley graphs based on [32] and [5]), perfect squares [28, 18], and other size distributions [35] or are only mildly explicit in the sense of having running time or parallel work that is $\mathrm{poly}(N)$ [26]. It was folklore that one can convert any sufficiently dense family of expanders (such as the aforementioned constructions) into a family of expanders of every size, and there were offhand remarks in the literature to this effect (see, e. g., [7], [19]), albeit without an analysis of expansion or explicitness. In Section 3, we give a self-contained description and proof of such a reduction, with an analysis in terms of both strong explicitness and spectral expansion. Subsequent to our work, Goldreich [20] has provided an analysis of the construction previously described in [19] in terms of vertex expansion and Alon [4] has given a stronger reduction that provides nearly Ramanujan expanders (i. e., ones with nearly optimal spectral expansion as a function of the degree) of every size N.

Many of our techniques are inspired by Cheng, Cheng, Liu, Peng, and Teng [14], who gave two algorithms for approximating random walks. One is a nearly linear time randomized algorithm for approximating random walks of *even* length and another works for all walk lengths r but has a running time that is quadratic in r, and so only yields a nearly linear time algorithm for r that is polylogarithmic in the size of the graph. In addition, [24] studied the problem of computing sparse spectral approximations of random walks but the running time in their work also has a quadratic dependence on r. We extend these results by giving a nearly linear time randomized algorithm for computing a spectral approximation to $I - M^r$ for *all* r. This is discussed in Section 5.

Subsequent work. Subsequent to [31], the conference version of this paper, Ahmadinejad et al. [1] strengthened Theorem 1.2 by providing a deterministic algorithm that, when given a graph G, two vertices s and t, and a positive integer r, estimates the probability that a random walk of length r in G started at s ends at t to within $\pm\varepsilon$ and uses space $O(\log(r\cdot N)\cdot\log\log(r\cdot N/\varepsilon))$. Furthermore, their result applies to the more general setting of *Eulerian digraphs*.[2] In undirected graphs, all nonzero conductances can be shown to be of magnitude at least $1/\mathrm{poly}(N)$ and can be expressed as a sum of at most $N^2/4$ random walk probabilities (entries of powers of the transition matrix). Consequently, with the same space bound the algorithm of [1] can estimate the conductance of any set S to within a multiplicative factor of $(1\pm 1/\mathrm{poly}(N))$. While the result of [1] is generally stronger than Theorem 1.2, with a better dependence on the approximation parameter ε, the algorithm and the analysis in our proof of Theorem 1.2 are considerably simpler than the corresponding items in [1]. The other results in our paper are not superseded by [1]. In particular, we believe that the derandomized graph product that we introduce to prove Theorem 1.2 is interesting in its own right. In addition, we fill a gap in the literature by extending the main result of [14] to compute sparse approximations of $I-M^r$ for *odd* r in nearly linear time, while their theorem works only for even r.

2 Preliminaries

2.1 Spectral graph theory

Given an undirected multigraph G the *Laplacian* of G is the symmetric matrix $D-A$, where D is the diagonal matrix of vertex degrees and A is the adjacency matrix of G. The *transition matrix* of the random walk on G is $T=AD^{-1}$. T_{ij} is the probability that a uniformly random edge from vertex j leads to vertex i (i. e., the number of edges between j and i divided by the degree of j). The *normalized Laplacian* of G is the symmetric matrix $I-M=D^{-1/2}(D-A)D^{-1/2}$. Note that when G is regular, D is a multiple of the identity and M and T are the same matrix: $M=D^{-1/2}AD^{-1/2}=AD^{-1}=T$. The *transition matrix of the r-step random walk* on G is T^r. For all probability distributions π, $T^r\pi$ is the distribution over vertices that results from picking a random vertex according to π and then running a random walk on G for r steps. The transition matrix of the r-step random walk on G is related to the normalized Laplacian in the following way:

$$I-M^r=D^{-1/2}(I-T^r)D^{1/2}.$$

For undirected multigraphs, the matrix $M=D^{-1/2}AD^{-1/2}$ has real eigenvalues between -1 and 1 and so $I-M^r$ has eigenvalues in $[0,2]$ and thus is positive semidefinite (PSD). The *spectral norm* of a real matrix M, denoted $\|M\|$, is the largest singular value of M. That is, the square root of the largest eigenvalue of M^TM. When M is symmetric, $\|M\|$ equals the largest eigenvalue of M in absolute value. For an undirected graph G with adjacency matrix A and a positive integer k, we write $k\cdot G$ to denote the graph with adjacency matrix $k\cdot A$, i. e., the multigraph G with all edges duplicated to have multiplicity k.

Given a symmetric matrix L, its *Moore-Penrose Pseudoinverse*, denoted $L^\dagger$, is the unique matrix with the same eigenvectors as L such that for each eigenvalue λ of L, the corresponding eigenvalue of $L^\dagger$ is

[2] Eulerian digraphs are directed graphs where the in-degree of each vertex equals its out-degree. Undirected graphs are a special case of Eulerian digraphs.

$1/\lambda$ if $\lambda \neq 0$ and 0 otherwise. When L is a Laplacian, we write $L^{\dagger/2}$ to denote the unique symmetric PSD matrix square root of the pseudoinverse of L.

To measure the approximation between graphs we use spectral approximation[3][38]:

Definition 2.1. Let $L, \widetilde{L} \in \mathbb{R}^{n \times n}$ be symmetric PSD matrices. We say that $\widetilde{L}$ *is an* ε-approximation *of* L (written $\widetilde{L} \approx_\varepsilon L$) if for all vectors $v \in \mathbb{R}^n$

$$(1-\varepsilon) \cdot v^T L v \leq v^T \widetilde{L} v \leq (1+\varepsilon) \cdot v^T L v.$$

Note that Definition 2.1 is not symmetric in L and $\widetilde{L}$. Spectral approximation can also be written in terms of the Loewner partial ordering of PSD matrices:

$$(1-\varepsilon) \cdot L \preceq \widetilde{L} \preceq (1+\varepsilon) \cdot L$$

where for two matrices A, B, we write $A \preceq B$ if $B - A$ is PSD. Spectral approximation has a number of useful properties listed in the following proposition.

Proposition 2.2. *If $W, X, Y, Z \in \mathbb{R}^{n \times n}$ are PSD symmetric matrices then:*

1. *if $X \approx_\varepsilon Y$ for $\varepsilon < 1$ then $Y \approx_{\varepsilon/(1-\varepsilon)} X$,*
2. *if $X \approx_{\varepsilon_1} Y$ and $Y \approx_{\varepsilon_2} Z$ then $X \approx_{\varepsilon_1+\varepsilon_2+\varepsilon_1 \cdot \varepsilon_2} Z$,*
3. *if $X \approx_\varepsilon Y$ and V is any $n \times n$ matrix then $V^T X V \approx_\varepsilon V^T Y V$,*
4. *if $X \approx_\varepsilon Y$ then $X + Z \approx_\varepsilon Y + Z$,*
5. *if $W \approx_{\varepsilon_1} X$ and $Y \approx_{\varepsilon_2} Z$ then $W + Y \approx_{\max\{\varepsilon_1, \varepsilon_2\}} X + Z$, and*
6. *if $X \approx_\varepsilon Y$ then $c \cdot X \approx_\varepsilon c \cdot Y$ for all nonnegative scalars c*

Following [41, 6, 3], we measure how well-connected a graph is using the second eigenvalue.

Definition 2.3. Let G be a regular undirected graph with transition matrix T. Define

$$\lambda(G) := \max_{\substack{v \perp \vec{1} \\ v \neq 0}} \frac{\|Tv\|}{\|v\|} = \text{2nd largest absolute value of the eigenvalues of } T \in [0,1].$$

$1 - \lambda(G)$ is called the *spectral gap* of G.

The smaller $\lambda(G)$, the faster a random walk on G converges to the uniform distribution. Graphs G with $\lambda(G)$ bounded away from 1 are called *expanders*. Expanders can equivalently be characterized as graphs that spectrally approximate the complete graph. This is formalized in the next lemma.

Lemma 2.4. *Let H be a c-regular undirected multigraph on n vertices with transition matrix T and let $J \in \mathbb{R}^{n \times n}$ be a matrix with $1/n$ in every entry (i. e., J is the transition matrix of the complete graph with a self-loop on every vertex). Then $\lambda(H) \leq \lambda$ if and only if $I - T \approx_\lambda I - J$.*

[3] In [30], we use an alternative definition of spectral approximation where $\widetilde{L} \approx_\varepsilon L$ if for all $v \in \mathbb{R}^n$, $\mathrm{e}^{-\varepsilon} \cdot v^T L v \leq v^T \widetilde{L} v \leq \mathrm{e}^{\varepsilon} \cdot v^T L v$. We find Definition 2.1 more convenient for this paper.

Proof. Let $v_1, v_2, \ldots, v_n$ be the orthonormal eigenvectors of $I-T$ where v_1 is the uniform distribution (the vector containing $1/n$ in every coordinate). Note that these are also eigenvectors of $I-J$ as $(I-J)v_1 = \vec{0}$ and $(I-J)v_i = v_i$ for all $i \neq 1$.

Let V be a matrix with $v_1, \ldots, v_n$ as columns. V is an orthogonal matrix so $V^T V = I$. From Proposition 2.2 Part 3, we have $I-T \approx_\lambda I-J$ if and only if $V^T(I-T)V \approx_\lambda V^T(I-J)V$. $V^T(I-T)V$ is a diagonal matrix with the eigenvalues $0, 1-\lambda_2, \ldots 1-\lambda_n$ of $I-T$ along the diagonal and $V^T(I-J)V$ is a diagonal matrix with the eigenvalues $0, 1, \ldots, 1$ of $I-J$ along the diagonal. Diagonal matrices spectrally approximate each other if and only if their diagonal entries satisfy the spectral inequalities, so $V^T(I-T)V \approx_\lambda V^T(I-J)V$ if and only if for all $i \in \{2, \ldots, n\}$

$$(1-\lambda) \cdot 1 \leq 1 - \lambda_i \leq (1+\lambda) \cdot 1.$$

The above holds if and only if $|\lambda_i| \leq \lambda$ for all $i \in \{2, \ldots, n\}$, which is equivalent to $\lambda(H) \leq \lambda$. □

In [15] Cohen, Kelner, Peebles, Peng, Rao, Sidford, and Vladu introduced a definition of spectral approximation for *asymmetric matrices*. Although the results in our paper only concern undirected graphs, some of our proofs use machinery from the theory of directed spectral approximation.

Definition 2.5 (Directed Laplacian [16, 15])**.** A matrix $L \in \mathbb{R}^{n \times n}$ is called a *directed Laplacian* if $L_{ij} \leq 0$ for all $i \neq j$ and $L_{ii} = -\sum_{j \neq i} L_{ji}$ for all $i \in [n]$. The associated directed graph has n vertices and an edge (i, j) of weight $-L_{ji}$ for all $i \neq j \in [n]$ with $L_{ji} \neq 0$.

Definition 2.6 (Asymmetric Matrix Approximation [15])**.** Let $\widetilde{L}$ and L be (possibly asymmetric) matrices such that $U = (L + L^T)/2$ is PSD. We say that $\widetilde{L}$ is a *directed ε-approximation* of L if:

1. $\ker(U) \subseteq \ker(\widetilde{L} - L) \cap \ker((\widetilde{L} - L)^T)$, and
2. $\left\| U^{\dagger/2} (\widetilde{L} - L) U^{\dagger/2} \right\|_2 \leq \varepsilon$.

Below we state some useful lemmas about directed spectral approximation. The first gives an equivalent formulation of Definition 2.6.

Lemma 2.7 ([15] Lemma 3.5)**.** *Let $L \in \mathbb{R}^{n \times n}$ be a (possibly asymmetric) matrix and let $U = (L + L^T)/2$. A matrix $\widetilde{L}$ is a directed ε-approximation of L if and only if for all vectors $x, y \in \mathbb{R}^n$*

$$x^T (\widetilde{L} - L) y \leq \frac{\varepsilon}{2} \cdot (x^T U x + y^T U y).$$

Lemma 2.8 ([15] Lemma 3.6)**.** *Suppose $\widetilde{L}$ is a directed ε-approximation of L and let $U = (L + L^T)/2$ and $\widetilde{U} = (\widetilde{L} + \widetilde{L}^T)/2$. Then $\widetilde{U} \approx_\varepsilon U$.*

Lemma 2.8 says that directed spectral approximation implies the usual notion from Definition 2.1 for "symmetrized" versions of the matrices L and $\widetilde{L}$. In fact, when the matrices L and $\widetilde{L}$ are both symmetric, the two definitions are equivalent:

Lemma 2.9. *Let $\widetilde{L}$ and L be symmetric PSD matrices. Then $\widetilde{L}$ is a directed ε-approximation of L if and only if $\widetilde{L} \approx_\varepsilon L$.*

Proof. The "only if" direction follows from Lemma 2.8 by observing that $U := (L+L^T)/2 = L$ and $\widetilde{U} := (\widetilde{L}+\widetilde{L}^T)/2 = \widetilde{L}$.

For the "if" direction, suppose $\widetilde{L} \approx_\varepsilon L$. Equivalently,

$$-\varepsilon \cdot L \preceq \widetilde{L} - L \preceq \varepsilon \cdot L.$$

Multiplying on the left and right by $L^{\dagger/2}$ preserves the ordering and yields

$$-\varepsilon \cdot L^{\dagger/2} L L^{\dagger/2} \preceq L^{\dagger/2}(\widetilde{L} - L) L^{\dagger/2} \preceq \varepsilon \cdot L^{\dagger/2} L L^{\dagger/2}.$$

For each nonzero eigenvalue λ of L, $L^{\dagger/2}$ has corresponding eigenvalue $1/\sqrt{\lambda}$ and hence $\left\| L^{\dagger/2} L L^{\dagger/2} \right\| \leq 1$. Since L is symmetric we have $U := (L+L^T)/2 = L$. Combining this with the above gives

$$\begin{aligned}
\left\| U^{\dagger/2}(\widetilde{L} - L) U^{\dagger/2} \right\| &= \left\| L^{\dagger/2}(\widetilde{L} - L) L^{\dagger/2} \right\| \\
&\leq \varepsilon \cdot \left\| L^{\dagger/2} L L^{\dagger/2} \right\| \\
&\leq \varepsilon.
\end{aligned}$$

Since $\widetilde{L} \approx_\varepsilon L$ by assumption, we must have $\ker(L) = \ker(\widetilde{L})$. It follows that $\ker(\widetilde{L} - L) = \ker(L)$ and $\ker((\widetilde{L} - L)^T) = \ker(L)$ since L and $\widetilde{L}$ are symmetric. Since $U = L$, we have

$$\begin{aligned}
\ker(U) &= \ker(L) \\
&= \ker(\widetilde{L} - L) \\
&= \ker(\widetilde{L} - L) \cap \ker((L - \widetilde{L})^T).
\end{aligned}$$

□

2.2 Space-bounded computation

We use a standard model of space-bounded computation where the machine $\mathcal{M}$ has a read-only input tape, a constant number of read/write work tapes, and a write-only output tape. If throughout every computation on inputs of length at most n, $\mathcal{M}$ uses at most $s(n)$ total tape cells on all the work tapes, we say $\mathcal{M}$ runs in space $s = s(n)$. Note that $\mathcal{M}$ may write more than s cells (in fact as many as $2^{O(s)}$) but the output tape is write-only. The following proposition describes the behavior of space complexity when space bounded algorithms are composed.

Proposition 2.10. *Let f_1, f_2 be functions that can be computed in space $s_1(n), s_2(n) \geq \log n$, respectively, and f_1 has output of length at most $\ell_1(n)$ on inputs of length n. Then $f_2 \circ f_1$ can be computed in space*

$$O(s_2(\ell_1(n)) + s_1(n)).$$

A proof of Proposition 2.10 can be found in [8] (Lemma 4.15).

2.3 Rotation maps

In the space-bounded setting, it is convenient to use local descriptions of graphs. Such descriptions allow us to navigate large graphs without loading them entirely into memory. For this we use *rotation maps*, functions that describe graphs through their neighbor relations. Rotation maps are defined for graphs with labeled edges as described in the following definition.

Definition 2.11 ([35])**.** A *two-way labeling* of an undirected multigraph $G = (V, E)$ with vertex degrees $(d_v)_{v \in V}$, is a labeling of the edges in G such that

1. every edge $(u, v) \in E$ has two labels: one in $[d_u]$ as an edge incident to u and one in $[d_v]$ as an edge incident to v;
2. for every vertex $v \in V$, the labels of the d_v edges incident to v are distinct.

In [36], two-way labelings are referred to as *undirected* two-way labelings. Note that every graph has a two-way labeling where each vertex "names" its neighbors uniquely in some canonical way based on the order they're represented in the input. We will describe multigraphs with two-way labelings using rotation maps:

Definition 2.12 ([35])**.** Let G be an undirected multigraph on n vertices with a two-way labeling. The *rotation map* Rot_G is defined as follows: $\mathrm{Rot}_G(v, i) = (w, j)$ if the i-th edge to vertex v leads to vertex w and this edge is the j-th edge incident to w.

We will use expanders that have efficiently computable rotation maps. We call such graphs *strongly explicit*. The usual definition of strong explicitness only refers to time complexity, but we will use it for both time and space.

Definition 2.13. A family of two-way labeled graphs $\mathcal{G} = \{G_{n,c}\}_{(n,c)}$, where $G_{n,c}$ is a c-regular graph on n vertices, is called *strongly explicit* if given n, c, a vertex $v \in [n]$ and an edge label $a \in [c]$, $\mathrm{Rot}_{G_{n,c}}(v, a)$ can be computed in time $\mathrm{poly}(\log(nc))$ and space $O(\log nc)$.

3 The derandomized product and expanders of all sizes

In this section we introduce our derandomized graph product. The derandomized product generalizes the *derandomized square* graph operation that was introduced by Rozenman and Vadhan [36] to give an alternative proof that UNDIRECTED S-T CONNECTIVITY is in **L**. Unlike the derandomized square, the derandomized product is defined for *irregular* graphs and produces a sparse approximation to the product of any two (potentially different) graphs with the same vertex degrees.

Here, by the "product" of two graphs G_0, G_1, we mean the reversible Markov chain with transitions defined as follows: from a vertex v, with probability $1/2$ take a random step on G_0 followed by a random step on G_1 and with probability $1/2$ take a random step on G_1 followed by a random step on G_0.

When $G_0 = G_1 = G$, this is the same as taking a 2-step random walk on G. Note, however, that when G is irregular, a 2-step random walk is *not* equivalent to doing a 1-step random walk on the graph G^2 whose edges correspond to paths of length 2 in G. Indeed, even the stationary distribution of the random walk on G^2 may be different than on G.[4] If A is the adjacency matrix of G, then G^2 has adjacency matrix A^2 and thus transition matrix $A^2D_2^{-1}$, where D_2 is a diagonal matrix whose v'th entry is the sum of the degrees of the neighbors of vertex v in G. In contrast, the two-step random walk on G has transition matrix $T^2 = (AD^{-1})^2$, where D is the diagonal matrix of degrees in G. Our goal in the derandomized product is to produce a relatively sparse graph whose 1-step random walk approximates the 2-step random walk on G.

The intuition behind the derandomized product is as follows: rather than build a graph with every such two-step walk, we use expander graphs to pick a pseudorandom subset of the walks. Specifically, we first pick $b \in \{0,1\}$ at random. Then, as before we take a truly random step from v to u in G_b. But for the second step, we don't use an arbitrary edge leaving u in $G_{\bar{b}}$, but rather correlate it to the edge on which we arrived at u using a c-regular expander on $\deg(u)$ vertices, where we assume that the vertex degrees in G_0 and G_1 are the same. When $c < \deg(u)$, the vertex degrees of the resulting two-step graph will be sparser than without derandomization. However using the pseudorandom properties of expander graphs, we can argue that the derandomized product is a good approximation of the true product. The combinatorial description of the derandomized product is as follows. For every vertex $u \in [n]$, we place a bipartite expander between the $\deg(u)$ neighbors of u in G_0 and the $\deg(u)$ neighbors of u in G_1.

Definition 3.1 (Derandomized Product)**.** Let G_0, G_1 be undirected multigraphs on n vertices with two-way labelings and identical vertex degrees $d_1, d_2, \ldots, d_n$. Let $\mathcal{H} = \{H_i\}$ be a family of two-way labeled, c-regular expanders of sizes including $d_1, \ldots, d_n$. The *derandomized product with respect to* $\mathcal{H}$, denoted $G_0 ⓟ_{\mathcal{H}} G_1$, is an undirected multigraph on n vertices with vertex degrees $2 \cdot c \cdot d_1, \ldots, 2 \cdot c \cdot d_n$ and rotation map $\mathrm{Rot}_{G_0 ⓟ_{\mathcal{H}} G_1}$ defined as follows: For $v_0 \in [n]$, $j_0 \in [d_{v_0}]$, $a_0 \in [c]$, and $b \in \{0,1\}$ we compute $\mathrm{Rot}_{G_0 ⓟ_{\mathcal{H}} G_1}(v_0, (j_0, a_0, b))$ as:

1. let $(v_1, j_1) = \mathrm{Rot}_{G_b}(v_0, j_0)$,
2. let $(j_2, a_1) = \mathrm{Rot}_{H_{d_{v_1}}}(j_1, a_0)$,
3. let $(v_2, j_3) = \mathrm{Rot}_{G_{\bar{b}}}(v_1, j_2)$,
4. output $(v_2, (j_3, a_1, \bar{b}))$,

where $\bar{b}$ denotes the bit-negation of b.

Note that when $G_0 = G_1$ the derandomized product generalizes the derandomized square [36] to irregular graphs, albeit with each edge duplicated twice. To see that $G_0 ⓟ_{\mathcal{H}} G_1$ is undirected, one can check that $\mathrm{Rot}_{G_0 ⓟ_{\mathcal{H}} G_1}(\mathrm{Rot}_{G_0 ⓟ_{\mathcal{H}} G_1}(v_0, (j_0, a_0, b))) = (v_0, (j_0, a_0, b)))$.

Note that Definition 3.1 requires that each vertex i has the same degree d_i in G_0 and G_1, ensuring that the random walks on G_0, G_1, and $G_0 ⓟ_{\mathcal{H}} G_1$ all have the same stationary distribution. This can be generalized to the case that there is an integer k such that for each vertex v with degree d_v in G_1, v has degree $k \cdot d_v$ in G_0. For this, we can duplicate each edge in G_1 k times to match the degrees of G_0 and then apply the derandomized product to the result. In such cases we abuse notation and write $G_0 ⓟ_{\mathcal{H}} G_1$ to mean $G_0 ⓟ_{\mathcal{H}} k \cdot G_1$.

[4] For example, let G be the graph on two vertices with one edge (u,v) connecting them and a single self-loop on u. Then $[2/3, 1/3]$ is the stationary distribution of G and $[3/5, 2/5]$ is the stationary distribution of G^2.

In [30] we showed that the derandomized square produces a spectral approximation to the true square. We now show that the derandomized product also spectrally approximates a natural graph product.

Theorem 3.2. *Let G_0, G_1 be undirected multigraphs on n vertices with two-way labelings, and normalized Laplacians $I - M_0$ and $I - M_1$. Let G_0 have vertex degrees $d_1, \ldots, d_n$ and G_1 have vertex degrees $d'_1, \ldots, d'_n$ where for all $i \in [n]$, $d_i = k \cdot d'_i$ for a positive integer k. Let $\mathcal{H} = \{H_i\}$ be a family of two-way labeled, c-regular expanders with $\lambda(H_i) \leq \lambda$ for all $H_i \in \mathcal{H}$, of sizes including $d_1, \ldots, d_n$. Let $I - \widetilde{M}$ be the normalized Laplacian of $\widetilde{G} = G_0 \circledcirc\!\!\!\!\text{p}_{\mathcal{H}} G_1$. Then*

$$I - \widetilde{M} \approx_\lambda I - \frac{1}{2} \cdot (M_0 M_1 + M_1 M_0).$$

Proof of Theorem 3.2. Note that $k \cdot G_1$ has the same transition matrix and normalized Laplacian as G_1. So we can replace G_1 with $k \cdot G_1$ and assume $k = 1$ without loss of generality.

Since G_0 and G_1 have the same vertex degrees, we can we write

$$I - \frac{1}{2} \cdot (M_0 M_1 + M_1 M_0) = I - D^{-1/2} \frac{1}{2} \cdot (T_0 T_1 + T_1 T_0) D^{1/2} \tag{3.1}$$

where T_0 and T_1 are the transition matrices of G_0 and G_1, respectively.

Following the proofs in [36] and [30], we can write the transition matrix for the random walk on $\widetilde{G}$ as $\widetilde{T} = \frac{1}{2} \cdot (P R_0 \widetilde{B} R_1 Q + P R_1 \widetilde{B} R_0 Q)$, where each matrix corresponds to a step in the definition of the derandomized product. The two terms correspond to $b = 0$ and $b = 1$ in the derandomized product and, setting $\bar{d} = \sum_{i \in [n]} d_i$:

- Q is a $\bar{d} \times n$ matrix that "lifts" a probability distribution over $[n]$ to one over $[\bar{d}]$ where the mass on each coordinate $i \in [n]$ is divided uniformly over the corresponding degree d_i. That is, $Q_{(u,i),v} = 1/d_i$ if $u = v$ and 0 otherwise where the rows of Q are ordered $(1,1), (1,2), \ldots, (1,d_1), (2,1), \ldots, (2,d_2), \ldots (n,1), \ldots, (n,d_n)$.

- R_0 and R_1 are the $\bar{d} \times \bar{d}$ symmetric permutation matrices corresponding to the rotation maps of G_0 and G_1, respectively. That is, entry $(u,i),(v,j)$ in R_a is 1 if $\mathrm{Rot}_{G_a}(u,i) = (v,j)$ and 0 otherwise for $a \in \{0,1\}$.

- $\widetilde{B}$ is a $\bar{d} \times \bar{d}$ symmetric block-diagonal matrix with n blocks where block i is the transition matrix for the random walk on $H_{d_i} \in \mathcal{H}$, the expander in our family with d_i vertices.

- $P = DQ^T$ is the $n \times \bar{d}$ matrix that maps any $\bar{d}$-vector to an n-vector by summing all the entries corresponding to edges incident to the same vertex in G_0 and G_1. This corresponds to projecting a distribution on $[\bar{d}]$ back down to a distribution over $[n]$. $P_{v,(u,i)} = 1$ if $u = v$ and 0 otherwise where the columns of P are ordered $(1,1), (1,2), \ldots, (1,d_1), (2,1), \ldots, (2,d_2), \ldots (n,1), \ldots, (n,d_n)$.

Likewise, we can write

$$(T_0 T_1 + T_1 T_0) = (P R_0 \widetilde{J} R_1 Q + P R_1 \widetilde{J} R_0 Q) \tag{3.2}$$

where $\widetilde{J}$ is a $\bar{d} \times \bar{d}$ symmetric block-diagonal matrix with n blocks where block i is J_i, the transition matrix for the complete graph on d_i vertices with a self-loop on every vertex. That is, every entry of J_i is $1/d_i$.

We will show that

$$I_{\bar{d}} - \frac{1}{2} \cdot (R_0 \bar{B} R_1 + R_1 \bar{B} R_0) \approx_\lambda I_{\bar{d}} - \frac{1}{2} \cdot (R_0 \bar{J} R_1 + R_1 \bar{J} R_0).$$

From this the theorem follows by multiplying by $D^{-1/2}P$ on the left and $(D^{-1/2}P)^T = QD^{1/2}$ on the right and applying Proposition 2.2 Part 3. Since $D^{-1/2}PQD^{1/2} = I_n$, the left-hand side becomes

$$\begin{aligned} I_n - D^{-1/2}\widetilde{T}D^{1/2} &= I_n - \widetilde{D}^{-1/2}\widetilde{T}\widetilde{D}^{1/2} \\ &= I_n - \widetilde{M} \end{aligned}$$

where $\widetilde{D} = 2 \cdot c \cdot D$ is the diagonal matrix of vertex degrees of $\widetilde{G}$. By Equation (3.1) and Equation (3.2), the right-hand side becomes $I_n - \frac{1}{2}(M_0 M_1 + M_1 M_0)$.

By Lemma 2.4, each graph in $\mathcal{H}$ is a λ-approximation of the complete graph on the same number of vertices. It follows that $I_{\bar{d}} - \widetilde{B} \approx_\lambda I_{\bar{d}} - \widetilde{J}$ because the quadratic form of a block diagonal matrix equals the sum of the quadratic forms of its blocks. By Lemma 2.9 and the fact that $I_d - \widetilde{J}$ is PSD, $I_{\bar{d}} - \widetilde{B}$ is also a directed λ-approximation of $I_{\bar{d}} - \widetilde{J}$. So for all vectors $x, y \in \mathbb{R}^{\bar{d}}$ we have

$$\begin{aligned} \left| x^T(\widetilde{B} - \widetilde{J})y \right| &\leq \frac{\lambda}{2} \cdot (x^T(I_{\bar{d}} - \widetilde{J})x + y^T(I_{\bar{d}} - \widetilde{J})y) \\ &\leq \frac{\lambda}{2} \cdot (x^T x + y^T y - 2x^T \widetilde{J} y). \end{aligned}$$

The first inequality uses Lemma 2.7. We can add the absolute values on the left-hand side since the right-hand side is always nonnegative ($I_d - \widetilde{J}$ is PSD) and invariant to swapping x with $-x$. The second inequality follows from the fact that $\widetilde{J}$ is PSD and so

$$0 \leq (x-y)^T \widetilde{J}(x-y) = x^T \widetilde{J} x + y^T \widetilde{J} y - 2 \cdot x^T \widetilde{J} y.$$

Fix $v \in \mathbb{R}^{\bar{d}}$ and set $x = R_0 v$ and $y = R_1 v$. Recall that R_0 and R_1 are symmetric permutation matrices and hence $R_0^2 = R_1^2 = I_{\bar{d}}$. Also note that for all square matrices A and vectors x, $x^T A x = x^T(A + A^T)x/2$. Combining these observations with the above gives

$$\begin{aligned} \left| v^T \left(\frac{1}{2} \cdot \left(R_0(\widetilde{B} - \widetilde{J})R_1 + R_1(\widetilde{B} - \widetilde{J})R_0 \right) \right) v \right| &= \left| v^T R_0 (\widetilde{B} - \widetilde{J}) R_1 v \right| \\ &\leq \frac{\lambda}{2} \cdot (v^T R_0^2 v + v^T R_1^2 v - 2 v^T R_0 \widetilde{J} R_1 v) \\ &= \lambda \cdot (v^T v - v^T R_0 \widetilde{J} R_1 v) \\ &= \lambda \cdot v^T \left(I - \frac{1}{2} \cdot \left(R_0 \widetilde{J} R_1 + R_1 \widetilde{J} R_0 \right) \right) v. \end{aligned}$$

Rearranging the above shows that

$$I_{\bar{d}} - \frac{1}{2} \cdot (R_0 \bar{B} R_1 + R_1 \bar{B} R_0) \approx_\lambda I_{\bar{d}} - \frac{1}{2} \cdot (R_0 \bar{J} R_1 + R_1 \bar{J} R_0),$$

which proves the theorem. □

Note that for a graph G with normalized Laplacian $I-M$ and transition matrix T, approximating $I-\frac{1}{2}\cdot(M_0M_1+M_1M_0)$ as in Theorem 3.2 for $M_0=M^{k_0}$ and $M_1=M^{k_1}$ gives a form of approximation to random walks of length k_1+k_2 on G, as

$$\begin{aligned} I-T^{k_1+k_2} &= D^{1/2}(I-M^{k_1+k_2})D^{-1/2} \\ &= I-\frac{1}{2}\cdot D^{1/2}(M_0M_1+M_1M_0)D^{-1/2}. \end{aligned}$$

To apply the derandomized product, we need an expander family $\mathcal{H}$ with sizes equal to all of the vertex degrees. However, existing constructions of strongly explicit expander families only give graphs of sizes that are subsets of $\mathbb{N}$ such as all powers of 2 or all perfect squares. In [36, 30] this was handled by adding self-loops to make the vertex degrees all equal and matching the sizes of expanders in explicit families. Adding self-loops was acceptable in those articles because it does not affect connectivity (the focus of [36]) or the Laplacian (the focus of [30]). However it does affect long random walks (our focus), so we cannot add self-loops. Instead, we show how to obtain strongly explicit expanders of all sizes. Our construction works by starting with a strongly explicit expander from one of the existing constructions and merging vertices to achieve any desired size:

Theorem 3.3. *There exists a family of strongly explicit expanders $\mathcal{H}$ such that for all $n>1$ and $\lambda\in(0,1)$ there is a $c=\mathrm{poly}(1/\lambda)$ and a c-regular graph $H_{n,c}\in\mathcal{H}$ on n vertices with $\lambda(H_{n,c})\leq\lambda$.*

Proof. Let H' be a c'-regular expander on $\hat{n}$ vertices such that $n\leq\hat{n}\leq 2n$, c' is a constant independent of n and $\lambda(H')\leq\lambda'<1/4$. H' can be constructed using already known strongly explicit constructions such as [18, 35] followed by squaring the graph a constant number of times to achieve $\lambda'<1/4$. We will construct H as follows: Pair off the first $(\hat{n}-n)$ vertices with the last $(\hat{n}-n)$ vertices in H' and merge each pair into a single vertex (which will then have degree $2\cdot c'$). To make the graph regular, add c' self-loops to all of the unpaired vertices. More precisely, given $u'\in[n]$ and $i'\in[c]=[2\cdot c']$ we compute $\mathrm{Rot}_H(u',i')$ as follows:

1. If $1\leq u'\leq\hat{n}-n$ [u' is a paired vertex]:
   (a) If $1\leq i'\leq c'$, let $u=u'$, $i=i'$ [u' is the first vertex in pair]
   (b) else let $u=\hat{n}-u'$, $i=i'-c'$ [u' is the second vertex in pair]
   (c) let $(v,j)=\mathrm{Rot}_{H'}(u,i)$
2. else (if $\hat{n}-n<u'\leq n$) [u' is an unpaired vertex]
   (a) If $1\leq i'\leq c'$, let $u=u'$, $i=i'$, and $(v,j)=\mathrm{Rot}_H(u,j)$ [original edge]
   (b) else let $(v,j)=(u',i')$ [new self-loop]
3. (a) If $v\leq n$, let $(v',j')=(v,j)$
   (b) else let $v'=\hat{n}-v$, $j'=j+c'$.
4. Output (v',j')

Notice from the description above that Rot_H can be computed in $O(\log n)$ space. Next we show that $\lambda(H)$ is bounded below 1 by a constant. The theorem then follows by taking the $O(\log 1/\lambda)$-th power of H to drive $\lambda(H)$ below λ. This gives the graph degree $c = \mathrm{poly}(1/\lambda)$. Since H has constant degree, edges in $H^{O(\log(1/\lambda))}$ can be represented in $O(\log(1/\lambda)) = O(\log c)$ space as a path of length $\log(1/\lambda)$ in H. Hence, computing neighbors in $H^{O(\log(1/\lambda))}$ can be done in the $O(\log n)$ space to compute neighbors in H plus the $O(\log n + \log c)$ space to follow a path of length $\log(1/\lambda)$ in H for a total of $O(\log(n \cdot c))$ space as desired.

Let A' be the adjacency matrix of H' and K' be the $\hat{n} \times \hat{n}$ all ones matrix. Since $\lambda(H') \leq \lambda'$, Lemma 2.4 implies that

$$\frac{1}{c'} \cdot (c' \cdot I - A') \approx_{\lambda'} \frac{1}{\hat{n}} \cdot (\hat{n} \cdot I - K').$$

Define B to be the $\hat{n} \times n$ matrix such that $B_{u',u} = 1$ if and only if vertex $u' \in V(H')$ was merged into vertex $u \in V(H)$ or vertex $u \in V(H')$ was not merged and is labeled vertex u' in H. That is, $B_{u',u} = 1$ if and only if $u = u'$ or $n \leq u = \hat{n} - u'$. Then the unnormalized Laplacian of the expander after the merging step is $B^T(c' \cdot I - A')B$. Adding self-loops to a graph does not change its Laplacian. So applying Proposition 2.2 parts 3 and 6 we get

$$L(H) = \frac{1}{2c'} \cdot B^T(c' \cdot I - A')B \approx_{\lambda'} \frac{1}{2\hat{n}} \cdot B^T(\hat{n} \cdot I - K)B.$$

Note that the righthand side is the normalized Laplacian of the graph U that results from starting with the complete graph on $\hat{n}$ vertices, merging the same pairs of vertices that are merged in H and adding $\hat{n}$ self-loops to all of the unmerged vertices for regularity.

We finish the proof by showing that $\lambda(U) \leq 1/2$ and thus H is a $(\lambda' + 1/2 + \lambda'/2)$-approximation of the complete graph by Proposition 2.2 Part 2 and Lemma 2.4. Recalling that $\lambda' < 1/4$ completes the proof.

U has at least $\hat{n}$ edges between every pair of vertices so we can write its transition matrix T_u as

$$T_u = \frac{1}{2} \cdot J_{\hat{n}} + \frac{1}{2} \cdot E$$

where $J_{\hat{n}}$ is the transition matrix of the complete graph on $\hat{n}$ vertices with self-loops on every vertex and E is the transition matrix for an $\hat{n}$-regular multigraph. Since E is the transition matrix of a regular graph, we have $\|E\| \leq 1$ and thus

$$\begin{aligned}
\lambda(U) &= \sup_{v \perp \vec{1}} \frac{\|T_u v\|}{\|v\|} \\
&\leq \sup_{v \perp \vec{1}} \frac{\frac{1}{2} \cdot (\|J_{\hat{n}} v\| + \|E v\|)}{\|v\|} \\
&\leq \frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 1,
\end{aligned}$$

which completes the proof. □

4 Main result

In this section we prove Theorem 1.4, our main result regarding space bounded computation of the normalized Laplacian of the r-step random walk on G.

Theorem 1.4 (restated). *There is a deterministic algorithm that given an undirected multigraph G on n vertices with normalized Laplacian $I-M$, a nonnegative integer r, and $\varepsilon > 0$, constructs an undirected multigraph $\widetilde{G}$ whose normalized Laplacian $\widetilde{L}$ is an ε-spectral approximation of $L = I - M^r$. That is, for all vectors $v \in \mathbb{R}^n$*

$$(1-\varepsilon)\cdot v^T L v \leq v^T \widetilde{L} v \leq (1+\varepsilon)\cdot v^T L v.$$

The algorithm runs in space

$$O(\log N + (\log r)\cdot \log(1/\varepsilon) + (\log r)\cdot \log\log r),$$

where N is the bit length of the input graph G.

The algorithm described below is inspired by techniques used in [14] to approximate random walks with a randomized algorithm in nearly linear time. Our analyses use ideas from the work of Cohen, Kelner, Peebles, Peng, Rao, Sidford, and Vladu on *directed* Laplacian system solvers even though all of the graphs we work with are undirected.

4.1 Algorithm description and proof overview

Let $I-M$ be the normalized Laplacian of our input and r be the target power. We will first describe an algorithm for computing $I-M^r$ without regard for space complexity and then convert it into a space-efficient approximation algorithm. The algorithm iteratively approximates larger and larger powers of M. On a given iteration, we will have computed $I-M^k$ for some $k<r$ and we use the following operations to increase k:

- Square: $I-M^k \to I-M^{2k}$,
- Plus one: $I-M^k \to I-\frac{1}{2}\cdot(M\cdot M^k + M^k\cdot M) = I-M^{k+1}$.

Interleaving these two operations appropriately can produce any power r of M, invoking each operation at most $\log_2 r$ times. To see this, let $b_z b_{z-1}\dots b_0$ be the bits of r in its binary representation where b_0 is the least significant bit and $b_z = 1$ is the most significant. We are given $I-M = I-M^{b_z}$. The algorithm will have z iterations and each one will add one more bit from most significant to least significant to the binary representation of the exponent. So after iteration i we will have $I-M^{b_z b_{z-1}\dots b_{z-i}}$.

For iterations $1,\dots,z$, we read the bits of r from b_{z-1} to b_0 one at a time. On each iteration we start with some power $I-M^k$. If the corresponding bit is a 0, we square to create $I-M^{2k}$ (which adds a 0 to the binary representation of the current exponent) and proceed to the next iteration. If the corresponding bit is a 1, we square and then invoke a plus one operation to produce $I-M^{2k+1}$ (which adds a 1 to the binary representation of the current exponent). After iteration z we will have $I-M^r$.

Implemented recursively, this algorithm has $\log_2 r$ levels of recursion and uses $O(\log N)$ space at each level for the matrix multiplications, where N is the bit length of the input graph. This results in total space

$O(\log r \cdot \log N)$, which is more than we want to use (cf. Theorem 1.4). We reduce the space complexity by replacing each square and plus one operation with the corresponding derandomized product, discussed in Section 3.

Theorem 3.2 says that the derandomized product produces spectral approximations to the square and the plus one operation. Since we apply these operations repeatedly on successive approximations, we need to maintain our ultimate approximation to a power of $I-M$. In other words, we need to show that given $\widetilde{G}$ such that

$$I-\widetilde{M} \approx_{\varepsilon} I-M^{k}$$

we have:

1. $I-\widetilde{M}^2 \approx_{\varepsilon} I-M^{2k}$,

2. $I-\frac{1}{2}\cdot(M\widetilde{M}+\widetilde{M}M) \approx_{\varepsilon} I-M^{k+1}$.

We prove these in Lemma 4.1 and Lemma 4.5. The transitive property of spectral approximation (Proposition 2.2 Part 2) will then complete the proof of spectral approximation.

We only know how to prove items 1 and 2 when M^k is PSD. This is problematic because M is not guaranteed to be PSD for arbitrary graphs and so M^k may only be PSD when k is even. Simple solutions like adding self-loops (to make the random walk lazy) are not available to us because loops may affect the random walk behavior in unpredictable ways. Another attempt would be to replace the plus one operation in the algorithm with a "plus two" operation

- Plus two: $I-M^k \rightarrow I-\frac{1}{2}\cdot(M^2\cdot M^k+M^k\cdot M^2) = I-M^{k+2}$.

Interleaving the square and plus two would preserve the positive semidefiniteness of the matrix we're approximating and can produce any even power of M. If r is odd, we could finish with one plus one operation, which will produce a spectral approximation because $I-M^{r-1}$ is PSD. A problem with this approach is that the derandomized product is defined only for unweighted multigraphs and M^2 may not correspond to an unweighted multigraph when G is irregular. (When G is regular, the graph G^2 consisting of paths of length 2 in G does have normalized Laplacian $I-M^2$.)

For this reason we begin the algorithm by constructing an unweighted multigraph G_0 whose normalized Laplacian $I-M_0$ approximates $I-M^2$ and where M_0 is PSD. We can then approximate any power $I-M_0^{r'}$ using the square and plus one operation and hence can approximate $I-M^r$ for any even r (see Lemma 4.6). For odd powers, we again can finish with a single plus one operation.

Our main algorithm is presented below. Our input is an undirected two-way labeled multigraph G with normalized Laplacian $I-M$, $\varepsilon \in (0,1)$, and $r = b_z b_{z-1} \ldots b_1 b_0$.

Algorithm 4.1
Input: G with normalized Laplacian $I-M$, $\varepsilon \in (0,1)$, $r = b_z b_{z-1} \dots b_1 b_0$
Output: G_z with normalized Laplacian $I-M_z$ such that $I-M_z \approx_\varepsilon I-M^r$

1. Set $\mu = \varepsilon/(32 \cdot z)$
2. Let $\mathcal{H}$ be family of expanders of every size such that $\lambda(H) \leq \mu$ for all $H \in \mathcal{H}$.
3. Construct G_0 such that $I-M_0 \approx_{\varepsilon/(16 \cdot z)} I-M^2$ and M_0 is PSD.
4. For i in $\{1,\dots,z-1\}$
   (a) If $b_{z-i} = 0$, $G_i = G_{i-1} ⓟ_{\mathcal{H}} G_{i-1}$
   (b) Else $G_i = (G_{i-1} ⓟ_{\mathcal{H}} G_{i-1}) ⓟ_{\mathcal{H}} G_0$
5. If $b_0 = 0$ (r even), $G_z = G_{z-1}$
6. Else (r is odd), $G_z = G_{z-1} ⓟ_{\mathcal{H}} G$
7. Output G_z

Note that each derandomized product multiplies every vertex degree by a factor of $2 \cdot c$. So the degrees of $G, G_0, \dots, G_z$ are all proportional to one another and the derandomized products in Algorithm 4.1 are well-defined.

4.2 Proof of main result

In this section we prove Theorem 1.4 by showing that Algorithm 4.1 yields a spectral approximation of our target power $I-M^r$ and can be implemented space-efficiently. First we show that our two operations, square and plus one, preserve spectral approximation.

Lemma 4.1 (Adapted from [14]). *Let N and $\widetilde{N}$ be symmetric matrices such that $I-\widetilde{N} \approx_\varepsilon I-N$ and N is PSD, then $I-\widetilde{N}^2 \approx_\varepsilon I-N^2$.*

The proof of Lemma 4.1 is adapted from Cheng, Cheng, Liu, Peng, and Teng [14], which uses ideas from the work of Miller and Peng [29]. We present these arguments here for the sake of completeness. We begin with three claims.

Claim 4.2. *Let N and $\widetilde{N}$ be symmetric matrices. If $I-\widetilde{N} \approx_\varepsilon I-N$ and N is PSD then $I+\widetilde{N} \approx_\varepsilon I+N$*

Proof. By definition, the hypothesis is equivalent to

$$-\varepsilon \cdot (I-N) \preceq (I-\widetilde{N}) - (I-N) \preceq \varepsilon \cdot (I-N),$$

which is equivalent to

$$-\varepsilon \cdot (I-N) \preceq (I+\widetilde{N}) - (I+N) \preceq \varepsilon \cdot (I-N).$$

Since N is PSD, we have $I-N \preceq I+N$. Thus we have

$$-\varepsilon \cdot (I+N) \preceq (I+N)-(I+\widetilde{N}) \preceq \varepsilon \cdot (I+N),$$

which is equivalent to $I+\widetilde{N} \approx_{\varepsilon} I+N$. □

Claim 4.3. *If N and $\widetilde{N}$ are symmetric matrices such that $I-\widetilde{N} \approx_{\varepsilon} I-N$ and $I+\widetilde{N} \approx_{\varepsilon} I+N$ then*

$$\begin{bmatrix} I & -\widetilde{N} \\ -\widetilde{N} & I \end{bmatrix} \approx_{\varepsilon} \begin{bmatrix} I & -N \\ -N & I \end{bmatrix}.$$

Proof. We follow the proof of Lemma 4.4 in [14]. For all vectors x,y and symmetric matrices U,

$$\begin{bmatrix} x \\ y \end{bmatrix}^T \begin{bmatrix} I & -U \\ -U & I \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{2} \cdot \left((x+y)^T (I-U)(x+y) + (x-y)^T (I+U)(x-y)\right).$$

The claim follows by taking $U=N$ and $U=\widetilde{N}$ and our assumptions that $I-\widetilde{N} \approx_{\varepsilon} I-N$ and $I+\widetilde{N} \approx_{\varepsilon} I+N$. □

Claim 4.4. *Let N be a symmetric matrix. Then*

$$x^T(I-N^2)x = \min_y \begin{bmatrix} x \\ y \end{bmatrix}^T \begin{bmatrix} I & -N \\ -N & I \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}.$$

Proof. This claim follows from Lemma B.2 in Miller and Peng [29], where it is stated in terms of the Schur complement. We modify the argument here without appealing to the Schur complement.

$$\begin{aligned} \begin{bmatrix} x \\ y \end{bmatrix}^T \begin{bmatrix} I & -N \\ -N & I \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} &= \|x\|^2 + \|y\|^2 - 2 \cdot \langle y, Nx \rangle \\ &\geq \|x\|^2 + \|y\|^2 - 2 \cdot \|y\| \cdot \|Nx\| \\ &\geq \|x\|^2 - \|Nx\|^2 \\ &= x^T(I-N^2)x. \end{aligned}$$

The first inequality follows from Cauchy-Schwarz and is tight if and only if $y = c \cdot Nx$ for some scalar c. The second inequality follows from the fact that $z^2 - 2 \cdot \|Nx\| \cdot z$ is minimized at $z = \|Nx\|$. So equality is achieved when $y = Nx$. □

Now we can prove Lemma 4.1.

Proof. Let

$$\mathbf{N} := \begin{bmatrix} I & -N \\ -N & I \end{bmatrix} \quad \text{and} \quad \widetilde{\mathbf{N}} := \begin{bmatrix} I & -\widetilde{N} \\ -\widetilde{N} & I \end{bmatrix}.$$

From our assumptions, it follows from Claim 4.2 that $I+\widetilde{N} \approx_\varepsilon I+N$ and hence that $\widetilde{\mathbf{N}} \approx_\varepsilon \mathbf{N}$ by Claim 4.3. Combining this with Claim 4.4 gives that for all vectors x,

$$\begin{aligned}(1-\varepsilon)\cdot x^T(I-N)x &\le (1-\varepsilon)\cdot \begin{bmatrix} x \\ \widetilde{N}x \end{bmatrix}^T \mathbf{N} \begin{bmatrix} x \\ \widetilde{N}x \end{bmatrix} \\ &\le \begin{bmatrix} x \\ \widetilde{N}x \end{bmatrix}^T \widetilde{\mathbf{N}} \begin{bmatrix} x \\ \widetilde{N}x \end{bmatrix} \\ &= x^T(I-\widetilde{N}^2)x.\end{aligned}$$

Similarly:

$$\begin{aligned}(1+\varepsilon)\cdot x^T(I-N^2)x &= (1+\varepsilon)\cdot \begin{bmatrix} x \\ Nx \end{bmatrix}^T \mathbf{N} \begin{bmatrix} x \\ Nx \end{bmatrix} \\ &\ge \begin{bmatrix} x \\ Nx \end{bmatrix}^T \widetilde{\mathbf{N}} \begin{bmatrix} x \\ Nx \end{bmatrix} \\ &\ge x^T(I-\widetilde{N}^2)x.\end{aligned}$$

So $I-\widetilde{N}^2 \approx_\varepsilon I-N^2$. □

Next we show that the plus one operation in our algorithm also preserves spectral approximation.

Lemma 4.5. *Let $\widetilde{N}$, N_1, and N_2 be symmetric matrices with spectral norm at most 1 and suppose that N_1 is PSD and commutes with N_2. If $I-\widetilde{N} \approx_\varepsilon I-N_1$ then*

$$I-\frac{1}{2}\cdot(\widetilde{N}N_2+N_2\widetilde{N}) \approx_\varepsilon I-N_2N_1.$$

Even though $I-(1/2)\cdot(\widetilde{N}N_2+N_2\widetilde{N})$ and $I-N_2N_1$ are both symmetric matrices (since N_1 and N_2 commute), the proof of Lemma 4.5 uses machinery from the theory of spectral approximation for *asymmetric* matrices. This is because the proof first shows that $I-N_2\widetilde{N}$ is an ε-approximation of $I-N_2N_1$ in the asymmetric sense (Definition 2.6). The asymmetric definition is necessary for this claim because $I-N_2\widetilde{N}$ can be an asymmetric matrix, so the usual notion of spectral approximation does not apply. We are not aware of any proofs of Lemma 4.5 that circumvent this application of asymmetric definitions.

Proof. Since $I-\widetilde{N} \approx_\varepsilon I-N_1$ and $I-N_1$ is PSD (since $\|N_1\| \le 1$), Lemma 2.7 and Lemma 2.9 say that for all $x,y \in \mathbb{R}^n$,

$$x^T(\widetilde{N}-N_1)y \le \frac{\varepsilon}{2}\cdot(x^T(I-N_1)x+y^T(I-N_1)y).$$

It follows (replacing x with N_2x) that for all vectors $x,y \in \mathbb{R}^n$,

$$x^T(N_2\widetilde{N}-N_2N_1)y \le \frac{\varepsilon}{2}\cdot(x^TN_2(I-N_1)N_2x+y^T(I-N_1)y).$$

We now claim that $N_2(I-N_1)N_2 \preceq I-N_1 \preceq I-N_2N_1$. If the claim is true then the above implies that for all vectors $x,y \in \mathbb{R}^n$

$$x^T(N_2\widetilde{N}-N_2N_1)y \leq \frac{\varepsilon}{2}\cdot(x^T(I-N_2N_1)x+y^T(I-N_2N_1)y),$$

which by Lemma 2.7 implies

$$\left\|(I-N_2N_1)^{\dagger/2}(N_2\widetilde{N}-N_2N_1)(I-N_2N_1)^{\dagger/2}\right\| \leq \varepsilon.$$

Since for all matrices U it is the case that $\|U+U^T\|/2 \leq \|U\|/2+\|U^T\|/2 = \|U\|$ we have

$$\left\|(I-N_2N_1)^{\dagger/2}\left(I-\frac{1}{2}\cdot(\widetilde{N}N_2+N_2\widetilde{N})-(I-N_2N_1)\right)(I-N_2N_1)^{\dagger/2}\right\| \leq \varepsilon.$$

Therefore $I-\frac{1}{2}\cdot(\widetilde{N}N_2+N_2\widetilde{N})$ is a directed ε-approximation of $I-N_2N_1$. Since these matrices are symmetric (N_2 and N_1 commute), Lemma 2.9 says that the approximation also holds in the undirected sense.

It remains to show that $N_2(I-N_1)N_2 \preceq I-N_1 \preceq I-N_2N_1$. Since N_1 and N_2 commute, they have the same eigenvectors [22]. So the inequalities reduce to inequalities of their corresponding eigenvalues. Let λ_1,λ_2 be eigenvalues corresponding to the same eigenvector of N_1,N_2, respectively. Since $\|N_1\|,\|N_2\| \leq 1$, we have $1-\lambda_1 \geq 0$ and $|\lambda_2| \leq 1$, so

$$\lambda_2\cdot(1-\lambda_1)\cdot\lambda_2 \leq 1-\lambda_1.$$

This proves that $N_2(I-N_1)N_2 \preceq I-N_1$. Also, since N_1 is PSD we have $\lambda_1 \geq 0$ and thus

$$1-\lambda_1 \leq 1-\lambda_2\cdot\lambda_1$$

because $|\lambda_2| \leq 1$. Thus $I-N_1 \preceq I-N_2N_1$ and the proof is complete. □

Setting $N_1 = M^k$ and $N_2 = M$ in Lemma 4.5 shows that the plus one operation preserves spectral approximation whenever M^k is PSD. Recall that the first step in Algorithm 4.1 is to construct a graph G_0 with normalized Laplacian $I-M_0$ such that M_0 is PSD and $I-M_0$ approximates $I-M^2$. We can then approximate $I-M_0^k$ for any k using squaring and plus one because M_0^k will always be PSD. The following Lemma says that $I-M_0^k$ spectrally approximates $I-M^{2k}$.

Lemma 4.6. *Let r be a positive integer with bit length $\ell(r)$ and A and B be symmetric PSD matrices with $\|A\|,\|B\| \leq 1$ such that $I-A \approx_\varepsilon I-B$ and $I-B \approx_\varepsilon I-A$ for $\varepsilon \leq 1/(2\cdot\ell(r))$. Then $I-A^r \approx_{2\cdot\varepsilon\cdot\ell(r)} I-B^r$.*

Proof. Let $b_zb_{z-1}\dots b_0$ be the bits in the binary representation of r with $b_z = 1$ being the most significant bit and b_0 being the least. For each $i \in \{0,\dots,z\}$, let r_i be the integer with binary representation $b_zb_{z-1}\dots b_{z-i}$.

The proof is by induction on i. As a base case, we have $I-A^{b_z} \approx_\varepsilon I-B^{b_z}$. We will show for the inductive step that for each $i \in \{1,\dots,z\}$, if $I-A^{r_{i-1}} \approx_{2\cdot(i-1)\cdot\varepsilon} I-B^{r_{i-1}}$ then $I-A^{r_i} \approx_{2\cdot i\cdot\varepsilon} I-B^{r_i}$. Assume $I-A^{r_{i-1}} \approx_{2\cdot(i-1)\cdot\varepsilon} I-B^{r_{i-1}}$. By Lemma 4.1, we have

$$I - A^{2 \cdot r_{i-1}} \approx_{2 \cdot (i-1) \cdot \varepsilon} I - B^{2 \cdot r_{i-1}}.$$

If $b_i = 0$ then $2 \cdot r_{i-1} = r_i$ so $I - A^{r_i} \approx_{2 \cdot (i-1) \cdot \varepsilon} I - B^{r_i}$. If $b_i = 1$, then by applying Lemma 4.5 we have

$$I - \frac{1}{2}(A^{2 \cdot r_{i-1}} B + B A^{2 \cdot r_{i-1}}) \approx_{2 \cdot (i-1) \cdot \varepsilon} I - B^{2 \cdot r_{i-1} + 1} = I - B^{r_i}.$$

Applying Lemma 4.5 again (recalling that $I - B \approx_\varepsilon I - A$ and A is PSD) gives

$$I - \frac{1}{2}(A^{2 \cdot r_{i-1}} B + B A^{2 \cdot r_{i-1}}) \approx_{\varepsilon} I - A^{2 \cdot r_{i-1} + 1} = I - A^{r_i}.$$

By Proposition 2.2 Part 1, this implies

$$I - A^{r_i} \approx_{\varepsilon/(1-\varepsilon)} I - \frac{1}{2}(A^{2 \cdot r_{i-1}} B + B A^{2 \cdot r_{i-1}}).$$

Putting this together with Proposition 2.2 Part 2 we have that $I - A^{r_i}$ is an $(\varepsilon/(1-\varepsilon) + 2 \cdot (i-1) \cdot \varepsilon + 2 \cdot (i-1) \cdot \varepsilon^2/(1-\varepsilon))$-approximation of $I - B^{r_i}$. Note that

$$\begin{aligned} 2 \cdot (i-1) \cdot \varepsilon + \frac{\varepsilon + 2 \cdot (i-1) \cdot \varepsilon^2}{1-\varepsilon} &\leq 2 \cdot (i-1) \cdot \varepsilon + \frac{2 \cdot \varepsilon - 2 \cdot \varepsilon^2}{1-\varepsilon} \\ &= 2 \cdot i \cdot \varepsilon \end{aligned}$$

where the first inequality follows from our assumption that $\varepsilon \leq 1/(2 \cdot \ell(r))$. So $I - A^{r_i} \approx_{2 \cdot i \cdot \varepsilon} I - B^{r_i}$, completing the inductive step. This shows that when $i = \ell(r)$, $I - A^r_{2 \cdot \ell(r) \cdot \varepsilon} I - B^r$, as desired. □

Now we can prove Theorem 1.4. We prove the theorem with three lemmas: Lemma 4.7 shows how to construct the graph G_0 needed in Algorithm 4.1, Lemma 4.8 argues that the algorithm produces a spectral approximation to $I - M^r$, and Lemma 4.9 shows that the algorithm can be implemented in space $O(\log N + (\log r) \cdot \log(1/\varepsilon) + (\log r) \cdot \log\log r)$.

4.2.1 Building G_0

Lemma 4.7. *There is an algorithm that takes an undirected, unweighted multigraph G with normalized Laplacian $I - M$ and a parameter $\varepsilon > 0$, and outputs a rotation map Rot_{G_0} for an undirected, unweighted multigraph G_0 with a two-way labeling and normalized Laplacian $I - M_0$ such that:*

1. *M_0 is PSD,*
2. *$I - M_0 \approx_\varepsilon I - M^2$, and*
3. *the algorithm uses space $O(\log N + \log(1/\varepsilon))$, where N is the bit length of the input graph G.*

Proof. Let $\delta = 1/\lceil 4/\varepsilon \rceil$ and $t = 1/\delta$, an integer. Let $\mathcal{H}$ be a family of c-regular expanders of every size from Theorem 3.3, such that for every $H \in \mathcal{H}$, $\lambda(H) \leq \delta$ (and hence $c = \mathrm{poly}(1/\delta)$).

Let $\widetilde{G} = G \textcircled{p}_{\mathcal{H}} G$ be the derandomized square of G with normalized Laplacian $I - \widetilde{M}$. Each vertex v in $\widetilde{G}$ has degree $\widetilde{d_v} = 2 \cdot c \cdot d_v$, where d_v is the degree of v in G. We construct G_0 as follows: duplicate every edge of $\widetilde{G}$ to have multiplicity t and then for each vertex v, add $\widetilde{d_v}$ self-loops. So for each vertex v in G_0, v has degree $(t+1) \cdot 2 \cdot c \cdot d_v$ and hence G_0 has the same stationary distribution as G. Note that we can write

$$M_0 = (t \cdot \widetilde{M} + I)/(t+1).$$

First we show that M_0 is PSD. From Theorem 3.2, we have $I - \widetilde{M} \approx_\delta I - M^2$, so $I - \widetilde{M} \preceq (1+\delta) \cdot (I - M^2) \preceq (1+\delta) \cdot I$, since M^2 is PSD. Thus $\widetilde{M} \succeq -\delta \cdot I$ and

$$M_0 \succeq \frac{t \cdot (-\delta \cdot I) + I}{t+1} \succeq 0.$$

Next we prove that $I - M_0 \approx_\varepsilon I - M^2$

$$\begin{aligned} I - M_0 &= (t/(t+1)) \cdot (I - \widetilde{M}) \\ &= \left(\frac{1}{1+\delta}\right) \cdot (I - \widetilde{M}) \\ &\preceq I - M^2. \end{aligned}$$

Observe that since $I - \widetilde{M} \approx_\delta I - M^2$, we also have

$$\begin{aligned} I - M_0 &= \left(\frac{1}{1+\delta}\right) \cdot (I - \widetilde{M}) \\ &\succeq \left(\frac{1-\delta}{1+\delta}\right) \cdot (I - M^2) \\ &\succeq (1-\varepsilon) \cdot (I - M^2). \end{aligned}$$

We can construct a two-way labeling of G in space $O(\log N)$ by arbitrarily numbering the edges incident to each vertex. Computing $\mathrm{Rot}_{\widetilde{G}}$ involves computing Rot_G twice and the rotation map of an expander in $\mathcal{H}$ once. For a given vertex degree d in G, Rot_{H_d} can be computed in space $O(\log(d \cdot c)) = O(\log N + \log(1/\varepsilon))$. Duplicating the edges and adding self-loops for Rot_{G_0} adds at most $O(\log N + \log(1/\varepsilon))$ overhead for a total of $O(\log N + \log(1/\varepsilon))$ space. $\square$

4.2.2 Proof of spectral approximation

Lemma 4.8. *Let G be an undirected multigraph with normalized Laplacian $I - M$, r be a positive integer and $\varepsilon \in (0,1)$. Let G_z be the output of Algorithm 4.1 with normalized Laplacian $I - M_z$. Then*

$$I - M_z \approx_\varepsilon I - M^r.$$

Proof. Let $b_z b_{z-1} \dots b_1 b_0$ be the binary representation of r. Recall that for the derandomized products in our algorithm we use a family of c-regular expanders $\mathcal{H}$ from Theorem 3.3 such that for every $H \in \mathcal{H}$, $\lambda(H) \le \mu = \varepsilon/(32 \cdot z)$ (and hence $c = \mathrm{poly}(1/\mu) = \mathrm{poly}((\log r)/\varepsilon)$).

We construct G_0 with normalized Laplacian $I - M_0$ as in Lemma 4.7 such that M_0 is PSD and $I - M_0 \approx_{\varepsilon/(16\cdot z)} I - M^2$. By Proposition 2.2 Part 1, and the fact that

$$\begin{aligned}\frac{\varepsilon/(16\cdot z)}{1-\varepsilon/(16\cdot z)} &= \frac{\varepsilon}{(16\cdot z)-\varepsilon}\\ &\le \frac{\varepsilon}{8\cdot z},\end{aligned}$$

we also have $I - M^2 \approx_{\varepsilon/(8\cdot z)} I - M_0$.

For each $i \in \{0, \dots z\}$ let r_i be the integer with binary representation $b_z b_{z-1} \dots b_{z-i}$ and let $I - M_i$ be the normalized Laplacian of G_i. We will prove by induction on i that G_i is a $(4\cdot\mu\cdot i)$-approximation to $I - M_0^{r_i}$. Thus, G_{z-1} is a $4\cdot\mu\cdot(z-1) \le \varepsilon/8$-approximation to $I - M_0^{r_{z-1}}$.

The base case is trivial since $r_0 = 1$. For the induction step, suppose that $I - M_{i-1} \approx_{4\cdot\mu\cdot(i-1)} I - M_0^{r_{z-i+1}}$. On iteration i, if $b_{z-i} = 0$, then $G_i = G_{i-1} ⓟ_{\mathcal{H}} G_{i-1}$. So we have

$$\begin{aligned} I - M_i &\approx_{\mu} I - M_{i-1}^2 \\ &\approx_{4\cdot\mu\cdot(i-1)} I - M_0^{2\cdot r_{i-1}} \\ &= I - M_0^{r_i} \end{aligned}$$

where the first approximation uses Theorem 3.2 and the second uses Lemma 4.1. By Proposition 2.2 Part 2 this implies that $I - M_i$ approximates $I - M_0^{r_i}$ with approximation factor

$$\mu + 4\cdot\mu\cdot(i-1) + 4\cdot\mu^2\cdot(i-1) \le 4\cdot\mu\cdot i$$

where we used the fact that $\mu < 1/(32\cdot(i-1))$.

If $b_{z-i} = 1$, $G_i = (G_{i-1} ⓟ_{\mathcal{H}} G_{i-1}) ⓟ_{\mathcal{H}} G_0$. Let $I - M_{\mathrm{ds}}$ be the normalized Laplacian of $G_{i-1} ⓟ_{\mathcal{H}} G_{i-1}$. By the analysis above, $I - M_{\mathrm{ds}}$ is a $(\mu + 4\cdot\mu\cdot(i-1) + 4\cdot\mu^2\cdot(i-1))$-approximation of $I - M_0^{2\cdot r_{i-1}}$. By Theorem 3.2 and Lemma 4.5 we have

$$\begin{aligned} I - M_i &\approx_{\mu} I - \frac{1}{2}\cdot(M_{\mathrm{ds}}M_0 + M_0 M_{\mathrm{ds}}) \\ &\approx_{\mu+4\cdot\mu\cdot(i-1)+4\cdot\mu^2\cdot(i-1)} I - M_0^{2\cdot r_{i-1}} M_0 \\ &= I - M_0^{r_i}. \end{aligned}$$

Applying Proposition 2.2 Part 2 and noting that $\mu \le 1/(32\cdot(i-1))$ we get

$$I - M_i \approx_{4\cdot\mu\cdot i} I - M_0^{r_i}.$$

So we conclude that $I - M_{z-1} \approx_{\varepsilon/8} I - M_0^{r_{z-1}}$. Furthermore, by Lemma 4.6 we have

$$I - M_0^{r_{z-1}} \approx_{\varepsilon/8} I - M^{2\cdot r_{z-1}}.$$

By Proposition 2.2 Part 2, and the fact that $\varepsilon \le 1$, this gives

$$I - M_{z-1} \approx_{\varepsilon/3} I - M^{2\cdot r_{z-1}}.$$

If $b_0 = 0$ then $2 \cdot r_{z-1} = r$ and we are done. If $b_0 = 1$ then we apply one more plus one operation using our original graph G to form $G_z = G_{z-1} ⓟ_{\mathcal{H}} G$ such that

$$\begin{aligned} I - M_z &\approx_\mu I - \frac{1}{2} \cdot (M_{z-1}M + MM_{z-1}) \\ &\approx_{\varepsilon/3} I - M^{2\cdot r_{z-1}+1} \\ &= I - M^r. \end{aligned}$$

Applying Proposition 2.2 Part 2 then gives $I - M_z \approx_\varepsilon I - M^r$. □

4.2.3 Analysis of space complexity

Lemma 4.9. *Algorithm 4.1 can be implemented so that given an undirected multigraph G, a positive integer r, and $\varepsilon \in (0,1)$, it computes its output G_z in space $O(\log N + (\log r) \cdot \log(1/\varepsilon) + (\log r) \cdot \log\log r)$, where N is the bit length of the input graph G.*

Proof. We show how to compute Rot_{G_z} in space $O(\log N + (\log r) \cdot \log(1/\varepsilon) + (\log r) \cdot \log\log r)$. Let $b_z b_{z-1} \dots b_0$ be the binary representation of r. Following Algorithm 4.1, G_0 is constructed with normalized Laplacian $I - M_0 \approx_{\varepsilon/(16\cdot z)} I - M^2$. From Lemma 4.7, we know Rot_{G_0} can be computed in space $O(\log N + \log(16 \cdot z/\varepsilon)) = O(\log N + \log(1/\varepsilon) + \log\log r)$. Let $d_1, \dots, d_n$ be the vertex degrees in G_0 and $d_{\max}$ be the maximum degree.

The algorithm is presented to have z iterations, where on iteration $i \in [z-1]$, if $b_{z-i} = 0$ the derandomized product is invoked once, and if $b_{z-i} = 1$, it is invoked twice. On iteration z it is either invoked once ($b_0 = 1$) or not at all ($b_0 = 0$). It will be simpler for us to think of each derandomized product happening in its own iteration. So we will consider $\tau = z + w = O(\log r)$ iterations where w is the number of ones in $b_{z-1}, \dots, b_0$. On iterations $1, \dots, z-1$, there are $z-1$ derandomized square operations and w plus one operations. The final iteration will either have a plus one operation with the graph G (if $b_0 = 1$) or no operation.

We copy the bits of r into memory and expand them into τ bits as follows: for $i \in \{1, \dots z-1\}$ if $b_{z-i} = 0$, record a 0 (corresponding to a derandomized square) and if $b_{z-i} = 1$, record a 0 followed by a 1 (corresponding to a derandomized square followed by a plus one operation). Finish by just recording b_z at the end. Now we have τ bits $t_1, \dots, t_\tau$ in memory where for $i < \tau$, $t_i = 0$ if the i-th derandomized product in our algorithm is a derandomized square and $t_i = 1$ if the i-th derandomized product is a plus one with the graph G_0. If $t_\tau = 0$, we do no derandomized product on the last iteration and if $t_\tau = 1$ we apply the plus one operation using G instead of G_0 as described in the algorithm.

We also re-number our graphs to be $G_1, \dots, G_\tau$ where G_i is the graph produced by following the derandomized products corresponding to $t_1, \dots, t_i$. For each $i \in [\tau]$ and $v \in [n]$, vertex v in graph G_i has degree $(2 \cdot c)^i \cdot d_v$ because each derandomized product multiplies every vertex degree by a factor of $2 \cdot c$.

Since our graphs can be irregular, the input to a rotation map may have a different length than its output. To simplify the space complexity analysis, when calling a rotation map, we will pad the edge labels to always have the same length as inputs and outputs to the rotation map. For each graph G_i, we pad its edge labels to have length $\ell_i = \lceil \log_2 d_{\max} \rceil + i \cdot \lceil \log_2(2 \cdot c) \rceil$.

Sublogarithmic-space complexity can depend on the model, so we will be explicit about the model we use. We compute the rotation map of each graph G_i on a multi-tape Turing machine with the following input/output conventions:

- Input Description:
  - Tape 1 (read-only): Contains the input G, r, and ε with the head at the leftmost position of the tape.
  - Tape 2 (read-write): Contains the input to the rotation map (v_0, k_0), where $v_0 \in [n]$ is a vertex of G_i, and k_0 is the label of an edge incident to v_0 padded to have total length ℓ_i. The tapehead is at the rightmost end of k_0. The rest of the tape may contain additional data.
  - Tape 3 (read-write): Contains the bits $t_1, \ldots, t_\tau$ with the head pointing at t_i.
  - Tapes 4+: (read-write): Blank worktapes with the head at the leftmost position.
- Output Description:
  - Tape 1: The head should be returned to the leftmost position.
  - Tape 2: In place of (v_0, k_0), it should contain $(v_2, k_2) = \mathrm{Rot}_{G_i}(v_0, k_0)$, where $v_2 \in [n]$, and k_2 is padded to have total length ℓ_i. The head should be at the rightmost position of k_2 and the rest of the tape should remain unchanged from its state at the beginning of the computation.
  - Tape 3: Contains the bits $t_1, \ldots, t_\tau$ with the head pointing at t_i.
  - Tapes 4+: (read-write): Are returned to the blank state with the heads at the leftmost position.

Let $\mathrm{Space}(G_i)$ be the space used on tapes other than tape 1 to compute Rot_{G_i}. We will show that $\mathrm{Space}(G_i) = \mathrm{Space}(G_{i-1}) + O(\log c)$. Recalling that $\mathrm{Space}(G_0) = O(\log N + \log(1/\varepsilon) + \log\log r)$ and unraveling the recursion gives

$$\begin{aligned}
\mathrm{Space}(G_z) &= O(\log N + \log(1/\varepsilon) + \log\log r + \tau \cdot \log c) \\
&= O(\log N + \log(1/\varepsilon) + \log\log r + \log r \cdot \log(\mathrm{poly}(\log r)/\varepsilon)) \\
&= O(\log N + (\log r) \cdot \log(1/\varepsilon) + (\log r) \cdot \log\log r)
\end{aligned}$$

as desired. Below we prove the recurrence on $\mathrm{Space}(G_i)$ by low level reasoning about rotation maps. While similar space complexity analysis has appeared before (see [34, 36, 30]), we first provide some intuition for how the space-efficient implementation of our algorithm works.

Our ultimate goal is, given a vertex v and edge label e in G_z, compute the e-th neighbor of v in G_z. If we can achieve this, we can compute the entire graph G_z by enumerating over vertices and edges. From the way we build G_z out of the derandomized product, neighbor relations in G_z are defined by neighbor relations in G_{z-1} as well as neighbor relations in an expander graph (in accordance with Definition 3.1). Likewise, neighbor relations in G_{z-1} are defined by neighbor relations in G_{z-2} as well as neighbor relations in an expander graph and so on. So we can recursively compute neighbors in graphs with larger subscripts by reducing those computations to neighbors in graphs with smaller subscripts along with computations on expander graphs.

A naive implementation of this recursion would use $O(\log n)$ space for each level of the $O(\log n)$ levels of recursion, for a total of $O(\log^2 n)$ space. The key advantage to using rotation maps and the structure inherent to the derandomized product, is that the space used to compute neighbors in the G_i in the algorithm can be reused. If we can compute neighbors in G_{i-1}, the only additional space needed to compute neighbors in G_i is the space to store an edge label from a graph in our expander family. This reasoning leads to the recurrence: $\mathrm{Space}(G_i) = \mathrm{Space}(G_{i-1}) + O(\log c)$.

To see how this operates at a low level with rotation maps, we begin with (v_0, k_0) on Tape 2 (possibly with additional data) and the tapehead at the far right of k_0. We parse k_0 into $k_0 = (j_0, a_0, b)$ where j_0 is an edge label in $[(2\cdot c)^{i-1}\cdot d_{v_0}]$ padded to have length ℓ_{i-1}, $a_0 \in [c]$, and $b \in \{0,1\}$.

Note that $G_i = G_{i-1} \textcircled{p}_{\mathcal{H}} G'$ where for $i \neq \tau$, we have $G' = G_{i-1}$ if $t_{i-1} = 0$ and $G' = G_0$ when $t_{i-1} = 1$. We compute Rot_{G_i} according to Definition 3.1. We move the head left to the rightmost position of j_0. If $b = 0$, we move the third tapehead to t_{i-1} and recursively compute $\mathrm{Rot}_{G_{i-1}}(v_0, j_0)$ so that Tape 2 now contains (v_1, j_1, a_0, b) (with j_1 padded to have the same length as j_0). The vertex v_1 in the graph G_{i-1} has degree $d' = (2\cdot c)^{i-1}\cdot d_{v_1}$ so we next compute $\mathrm{Rot}_{H_{d'}}(j_1, a_0)$ so that (v_1, j_2, a_1, b) is on the tape. Finally we compute $\mathrm{Rot}_{G'}(v_1, j_2)$ and flip b to finish with $(v_2, j_3, a_1, \bar{b})$ on the second tape. We then move the third tapehead to t_i. If $b = 1$ then we just swap the roles of G_{i-1} and G' above.

So computing Rot_{G_i} involves computing the rotation maps of G_{i-1}, $H_{d'}$, and G' each once. Note that each of the rotation map evaluations occur in succession and can therefore reuse the same space. Clearly $\mathrm{Space}(G') \leq \mathrm{Space}(G_{i-1})$ because either $G' = G_{i-1}$ or G' is either G_0 or G, both of whose rotation maps are subroutines in computing $\mathrm{Rot}_{G_{i-1}}$. Computing $\mathrm{Rot}_{H_{d'}}$ adds an overhead of at most $O(\log c)$ space to store the additional edge label a_0 and the bit b. So we can compute the rotation map of G_τ in space $O(\log N + (\log r)\cdot\log(1/\varepsilon) + (\log r)\cdot\log\log r)$. □

5 Corollaries

5.1 Random walks

Our algorithm immediately implies Theorem 1.2, which we prove below.

Theorem 1.2 Restated. *There is a deterministic algorithm that given an undirected multigraph G on n vertices, a positive integer r, a set of vertices S, and $\varepsilon > 0$, computes a number $\widetilde{\Phi}$ such that*

$$(1-\varepsilon)\cdot\Phi_r(S) \leq \widetilde{\Phi} \leq (1+\varepsilon)\cdot\Phi_r(S)$$

and runs in space $O(\log N + (\log r)\cdot\log(1/\varepsilon) + (\log r)\cdot\log\log r)$, where N is the bit length of the input graph G (the length of the description of G as a list of edges, in a binary encoding).

Proof of Theorem 1.2. Let D be the diagonal degree matrix and $I - M$ be the normalized Laplacian of G. Let $v = D^{1/2}e_S$ where e_S is the characteristic vector of the set S. Let d_S be the sum of the degrees of vertices in S. Then using the fact that $I - M^r = D^{-1/2}(I - T^r)D^{1/2}$ where T is the transition matrix of G

gives:

$$\begin{aligned}\frac{1}{d_S} \cdot v^T (I - M^r) v &= \frac{1}{d_S} \cdot e_S^T D^{1/2} D^{-1/2} (I - T^r) D^{1/2} D^{1/2} e_S \\ &= \frac{1}{d_S} \cdot e_S^T D e_S - e_S^T (T^r (D e_S / d_S)) \\ &= 1 - \Pr[V_r \in S | V_0 \in S] \\ &= \Phi_r(S)\end{aligned}$$

where the penultimate equality follows from the fact that De_S/d_S is the probability distribution over vertices in S where each vertex has mass proportional to its degree, i. e., the probability distribution $V_0 \| (V_0 \in S)$. Multiplying this distribution by T^r gives the distribution of $V_r \| (V_0 \in S)$. Multiplying this resulting distribution on the left by e_S^T, sums up the probabilities over vertices in S, which gives the probability that our random walk ends in S.

From Theorem 1.4, we can compute a matrix $\widetilde{L}$ such that $\widetilde{L} \approx_\varepsilon I - M^r$ in space $O(\log N + (\log r) \cdot \log(1/\varepsilon) + (\log r) \cdot \log\log r)$. It follows from Proposition 2.2, Part 6 and the definition of spectral approximation that

$$(1 - \varepsilon) \cdot \Phi_r(S) \leq \frac{1}{d_S} \cdot v^T \widetilde{L} v \leq (1 + \varepsilon) \cdot \Phi_r(S). \qquad \square$$

Our algorithm also implies an algorithm for approximating random walk matrix polynomials.

Definition 5.1 (Random walk matrix polynomials)**.** Let G be an undirected multigraph with Laplacian $D - A$ and let α be a vector of nonnegative scalars $\alpha = (\alpha_1, \ldots, \alpha_\ell)$ such that $\sum_{i \in [\ell]} \alpha_i = 1$. The normalized *$\ell$-degree random walk matrix polynomial* of G with respect to α is defined to be:

$$\begin{aligned}L_\alpha(G) &:= I - \sum_{r=1}^{\ell} \alpha_r \cdot M^r \\ &= D^{-1/2} \left(I - \sum_{r=1}^{\ell} \alpha_r \cdot T^r \right) D^{1/2},\end{aligned}$$

where $M = D^{-1/2} A D^{-1/2}$ and $T = AD^{-1}$

Random walk matrix polynomials are Laplacian matrices that arise in algorithms for approximating q-th roots of symmetric diagonally dominant matrices and efficient sampling from Gaussian graphical models [27, 14, 13].

Corollary 5.2. *There is a deterministic algorithm that given an undirected multigraph G, a positive integer ℓ, a vector of nonnegative scalars $\alpha = (\alpha_1, \ldots, \alpha_\ell)$ that sum to 1, and $\varepsilon > 0$, computes an ε-approximation to $L_\alpha(G)$ and runs in space* $O(\log N + (\log r) \cdot \log(1/\varepsilon) + (\log r) \cdot \log\log r)$.

Proof. For each $r \in [\ell]$, let L_r be an ε-approximation of $I - M^r$. Applying Proposition 2.2, Parts 5 and 6 gives

$$\begin{aligned}\sum_{r=1}^{\ell} \alpha_r \cdot L_r \approx_\varepsilon \sum_{r=1}^{\ell} \alpha_r \cdot I - \alpha_r \cdot M^r \\ = I - \sum_{r=1}^{\ell} \alpha_r \cdot M^r \\ = L_\alpha(G).\end{aligned}$$

Our algorithm can compute each L_r in space $O(\log N + (\log r) \cdot \log(1/\varepsilon) + (\log r) \cdot \log\log r)$. Multiplication and iterated addition can both be computed in $O(\log N)$ space where N is the input bit length [10, 21]. So by the composition of space bounded algorithms (Proposition 2.10), $L_\alpha(G)$ can be approximated in space $O(\log N + (\log r) \cdot \log(1/\varepsilon) + (\log r) \cdot \log\log r)$. □

5.2 Odd-length walks in nearly linear time

Our approach to approximating odd-length walks deterministically and space-efficiently also leads to a new result in the context of nearly linear-time (randomized) spectral sparsification algorithms. Specifically, we extend the following Theorem of Cheng, Cheng, Liu, Peng, and Teng [14].

Theorem 5.3 ([14])**.** *There is a randomized algorithm that given an undirected weighted graph G with n vertices, m edges, and normalized Laplacian $I - M$,* even *integer r, and $\varepsilon > 0$ constructs an undirected weighted graph $\widetilde{G}$ with normalized Laplacian $\widetilde{L}$ containing $O(n \log n/\varepsilon^2)$ non-zero entries, in time $O(m \cdot \log^3 n \cdot \log^5 r/\varepsilon^4)$, such that $\widetilde{L} \approx_\varepsilon I - M^r$ with high probability.*

Our approach to approximating odd-length walks can be used to extend Theorem 5.3 to odd r.

Corollary 5.4. *There is a randomized algorithm that given an undirected weighted graph G with n vertices, m edges, and normalized Laplacian $I - M$,* odd *integer r, and $\varepsilon > 0$ constructs an undirected weighted graph $\widetilde{G}$ with normalized Laplacian $\widetilde{L}$ containing $O(n \log n/\varepsilon^2)$ non-zero entries, in time $O(m \cdot \log^3 n \cdot \log^5 r/\varepsilon^4)$, such that $\widetilde{L} \approx_\varepsilon I - M^r$ with high probability.*

Our proof of Corollary 5.4 uses Theorem 5.3 as a black box. So in fact, given G with normalized Laplacian $I - M$ and any graph $\widetilde{G}$ whose normalized Laplacian approximates $I - M^r$ for even r, we can produce an approximation to $I - M^{r+1}$ in time nearly linear in the sparsities of G and $\widetilde{G}$. To prove the corollary, we use the same method used in [33] and [15] for sparsifying two-step walks on undirected and directed graphs, respectively. The idea is that the graphs constructed from two-step walks can be decomposed into the union of *product graphs*: graphs whose adjacency matrices have the form xy^T for vectors $x, y \in \mathbb{R}^n$. We use the following fact from [15] that says that product graphs can be sparsified in time that is nearly-linear in the number of non-zero entries of x and y rather than the number of non-zero entries in xy^T, which may be much larger.

Lemma 5.5 (Adapted from [15] Lemma 3.18)**.** *Let x, y be non-negative vectors with $\|x\|_1 = \|y\|_1 = r$ and let $\varepsilon \in (0,1)$. Furthermore, let s denote the total number of non-zero entries in x and y and let $L = \operatorname{diag}(y) - \frac{1}{r} \cdot xy^T$. Then there is an algorithm that in time $O(s \cdot \log s/\varepsilon^2)$ computes a matrix $\widetilde{L}$ with $O(s \cdot \log s/\varepsilon^2)$ non-zeros such that $\widetilde{L}$ is a directed ε-approximation of L with high probability.*

After using Lemma 5.5 to sparsify each product graph in our decomposition, we then apply an additional round of graph sparsification.

Lemma 5.6 ([25])**.** *Given an undirected graph G with n vertices, m edges, and Laplacian L and $\varepsilon > 0$, there is an algorithm that computes a graph $\widetilde{G}$ with Laplacian $\widetilde{L}$ containing $O(n \cdot \log n/\varepsilon^2)$ non-zero entries in time $O(m \cdot \log^2 n/\varepsilon^2)$ such that $\widetilde{L} \approx_\varepsilon L$ with high probability.*

Now we can prove Corollary 5.4

Proof of Corollary 5.4. Theorem 5.3 says that we can compute a graph $\widetilde{G}$ with normalized Laplacian $I - \widetilde{M}$ with $O(n \log n/\varepsilon^2)$ non-zero entries, in time $O(m \cdot \log^3 n \cdot \log^5 r/\varepsilon^4)$, such that $I - \widetilde{M} \approx_{\varepsilon/8} I - M^{r-1}$ with high probability. By Lemma 4.5 we have

$$I - \frac{1}{2} \cdot (\widetilde{M}M + M\widetilde{M}) \approx_{\varepsilon/8} I - M^r. \tag{5.1}$$

Our goal is to sparsify the lefthand side. Note that since $I - \widetilde{M}$ spectrally approximates $I - M^{r-1}$, the corresponding graphs must have the same stationary distribution and hence proportional vertex degrees. In other words there is a number k such that for all vertices $v \in [n]$ we have $\deg_{\widetilde{G}}(v) = k \cdot \deg_G(v)$. We will think of the graph that adds one step to our walk as $k \cdot G$ rather than G because $k \cdot G$ and $\widetilde{G}$ have the same degrees and the normalized Laplacian of $k \cdot G$ is the same as the normalized Laplacian of G.

Let A and $\widetilde{A}$ be the adjacency matrices of $k \cdot G$ and $\widetilde{G}$, respectively and let D be the diagonal matrix of vertex degrees. Let $Q = D - AD^{-1}\widetilde{A}$ and note that Q is the Laplacian of a weighted directed graph. We will show how to compute a sparse directed approximation of Q and use this to show how to compute a sparse approximation to the lefthand side of Equation (5.1). Our approach is inspired by similar arguments from [33, 15]. We decompose Q into n product graphs as follows. For each $i \in [n]$ let

$$Q_i = \operatorname{diag}(\widetilde{A}_{i,:}) - \frac{1}{D_{i,i}} \cdot A_{:,i}\widetilde{A}_{i,:}^T$$

where $\widetilde{A}_{i,:}$ and $A_{:,i}$ denote the i-th row of $\widetilde{A}$ and the i-th column of A, respectively. Observe that Q_i is a directed Laplacian of a bipartite graph between the neighbors of vertex i in $k \cdot G$ and the neighbors of i in $\widetilde{G}$ and that $Q = \sum_{i \in [n]} Q_i$. Furthermore, each Q_i is a product graph and hence can be sparsified using Lemma 5.5. Set $x_i = A_{:,i}$, $y_i = \widetilde{A}_{i,:}$, $r_i = D_{i,i}$, and let s_i be the total number of non-zero entries in x and y. Note that $\|x_i\|_1 = \|y_i\|_1 = r_i$ because $k \cdot G$ and $\widetilde{G}$ have the same vertex degrees. By Lemma 5.5, for each $i \in [n]$ we can compute a directed $\varepsilon/8$-approximation $\widetilde{Q}_i$ of Q_i containing $O(s_i \cdot \log s_i/\varepsilon^2)$ entries in time $O(s_i \cdot \log s_i/\varepsilon^2)$. Applying the lemma to each Q_i yields $\widetilde{Q} = \sum_{i \in [n]} \widetilde{Q}_i$, which contains $O(m \cdot \log m/\varepsilon^2)$ non-zero entries and can be computed in time $O(m \cdot \log m/\varepsilon^2)$ because $\sum_{i \in [n]} s_i = O(m)$. By Lemma 2.8 we have

$$\frac{1}{2} \cdot (\widetilde{Q}_i + \widetilde{Q}_i^T) \approx_{\varepsilon/8} \frac{1}{2} \cdot (Q_i + Q_i^T)$$

for all $i \in [n]$ with high probability. It follows from Proposition 2.2 Part 5 that

$$\begin{aligned}\frac{1}{2}\cdot(\widetilde{Q}+\widetilde{Q}^T) &= \frac{1}{2}\cdot\sum_{i\in[n]}(\widetilde{Q}_i+\widetilde{Q}_i^T)\\ &\approx_{\varepsilon/8}\frac{1}{2}\cdot\sum_{i\in[n]}(Q_i+Q_i^T)\\ &=\frac{1}{2}\cdot(Q+Q^T)\end{aligned}$$

with high probability. From Proposition 2.2 Part 3, we then get

$$\begin{aligned}D^{-1/2}\frac{1}{2}\cdot(\widetilde{Q}+\widetilde{Q}^T)D^{-1/2} &\approx_{\varepsilon/8} D^{-1/2}\frac{1}{2}\cdot(Q+Q^T)D^{-1/2}\\ &= I-\frac{1}{2}\cdot(\widetilde{M}M+M\widetilde{M})\end{aligned}$$

with high probability. Applying Lemma 5.6 we can re-sparsify the graph corresponding to $D^{-1/2}\frac{1}{2}\cdot(\widetilde{Q}+\widetilde{Q}^T)D^{-1/2}$ to produce a graph G' whose normalized Laplacian $I-M'$ has $O(n\cdot\log n/\varepsilon^2)$ non-zero entries and $I-M'\approx_{\varepsilon/8}D^{-1/2}\frac{1}{2}\cdot(\widetilde{Q}+\widetilde{Q}^T)D^{-1/2}$ with high probability. This takes additional time $O(m\cdot\log^2 n/\varepsilon^2)$ due to Theorem 1.1 of [25]. Applying Proposition 2.2 Part 2 twice we get that $I-M'\approx_\varepsilon I-M^r$ and the total running time for the procedure was $O(m\cdot\log^3 n\cdot\log^5 r/\varepsilon^4)$. □

References

[1] AMIRMAHDI AHMADINEJAD, JONATHAN A. KELNER, JACK MURTAGH, JOHN PEEBLES, AARON SIDFORD, AND SALIL VADHAN: High-precision estimation of random walks in small space. In *Proc. 61st FOCS*, pp. 1295–1306. IEEE Comp. Soc., 2020. [doi:10.1109/FOCS46700.2020.00123] 5

[2] ROMAS ALELIUNAS, RICHARD M. KARP, RICHARD J. LIPTON, LÁSZLÓ LOVÁSZ, AND CHARLES RACKOFF: Random walks, universal traversal sequences, and the complexity of maze problems. In *Proc. 20th FOCS*, pp. 218–223. IEEE Comp. Soc., 1979. [doi:10.1109/SFCS.1979.34] 2

[3] NOGA ALON: Eigenvalues and expanders. *Combinatorica*, 6(2):83–96, 1986. Preliminary version in STOC'84. [doi:10.1007/BF02579166] 6

[4] NOGA ALON: Explicit expanders of every degree and size. *Combinatorica*, 2021. [doi:10.1007/s00493-020-4429-x, arXiv:2003.11673] 4

[5] NOGA ALON, ODED GOLDREICH, JOHAN HÅSTAD, AND RENÉ PERALTA: Simple constructions of almost k-wise independent random variables. *Random Struct. Algor.*, 3(3):289–304, 1992. See also addendum in vol. 4(1), 1993, pp. 199–120. [doi:10.1002/rsa.3240030308] 4

[6] NOGA ALON AND VITALY D. MILMAN: λ_1, isoperimetric inequalities for graphs, and superconcentrators. *J. Combin. Theory–B*, 38(1):73–88, 1985. [doi:10.1016/0095-8956(85)90092-9] 6

[7] NOGA ALON, ODED SCHWARTZ, AND ASAF SHAPIRA: An elementary construction of constant-degree expanders. *Combin. Probab. Comput.*, 17(3):319–327, 2008. [doi:10.1017/S0963548307008851] 4

[8] SANJEEV ARORA AND BOAZ BARAK: *Computational Complexity: A modern approach*. Cambridge Univ. Press, 2009. 8

[9] JOSHUA BATSON, DANIEL A. SPIELMAN, NIKHIL SRIVASTAVA, AND SHANG-HUA TENG: Spectral sparsification of graphs: Theory and algorithms. *Comm. ACM*, 56(8):87–94, 2013. [doi:10.1145/2492007.2492029] 3

[10] PAUL W. BEAME, STEPHEN A. COOK, AND H. JAMES HOOVER: Log depth circuits for division and related problems. *SIAM J. Comput.*, 15(4):994–1003, 1986. [doi:10.1137/0215070] 28

[11] MARK BRAVERMAN, ANUP RAO, RAN RAZ, AND AMIR YEHUDAYOFF: Pseudorandom generators for regular branching programs. *SIAM J. Comput.*, 43(3):973–986, 2014. Preliminary version in FOCS'10. [doi:10.1137/120875673] 3

[12] JOSHUA BRODY AND ELAD VERBIN: The coin problem and pseudorandomness for branching programs. In *Proc. 51st FOCS*, pp. 30–39. IEEE Comp. Soc., 2010. [doi:10.1109/FOCS.2010.10] 3

[13] DEHUA CHENG, YU CHENG, YAN LIU, RICHARD PENG, AND SHANG-HUA TENG: Scalable parallel factorizations of SDD matrices and efficient sampling for Gaussian graphical models. 2014. [arXiv:1410.5392] 27

[14] DEHUA CHENG, YU CHENG, YAN LIU, RICHARD PENG, AND SHANG-HUA TENG: Spectral sparsification of random-walk matrix polynomials. 2015. [arXiv:1502.03496] 3, 4, 5, 15, 17, 18, 27, 28

[15] MICHAEL B. COHEN, JONATHAN A. KELNER, JOHN PEEBLES, RICHARD PENG, ANUP RAO, AARON SIDFORD, AND ADRIAN VLADU: Almost-linear-time algorithms for Markov chains and new spectral primitives for directed graphs. In *Proc. 49th STOC*, pp. 410–419. ACM Press, 2017. [doi:10.1145/3055399.3055463] 3, 4, 7, 28, 29

[16] MICHAEL B. COHEN, JONATHAN A. KELNER, JOHN PEEBLES, RICHARD PENG, AARON SIDFORD, AND ADRIAN VLADU: Faster algorithms for computing the stationary distribution, simulating random walks, and more. In *Proc. 57th FOCS*, pp. 583–592. IEEE Comp. Soc., 2016. [doi:10.1109/FOCS.2016.69] 3, 7

[17] ANINDYA DE: Pseudorandomness for permutation and regular branching programs. In *Proc. 26th IEEE Conf. on Comput. Complexity (CCC'11)*, pp. 221–231. IEEE Comp. Soc., 2011. [doi:10.1109/CCC.2011.23] 3

[18] OFER GABBER AND ZVI GALIL: Explicit constructions of linear-sized superconcentrators. *J. Comput. System Sci.*, 22(3):407–420, 1981. Preliminary version in FOCS'79. [doi:10.1016/0022-0000(81)90040-4] 4, 13

[19] ODED GOLDREICH: *Computational complexity: A conceptual perspective*. Cambridge Univ. Press, 2008. [doi:10.1017/CBO9780511804106] 4

[20] ODED GOLDREICH: On constructing expanders for any number of vertices. In ODED GOLDREICH, editor, *Computational Complexity and Property Testing*, volume 12050 of *LNCS*, pp. 374–379. Springer, 2020. Preliminary version in Author's home page. [doi:10.1007/978-3-030-43662-9_21] 4

[21] WILLIAM HESSE, ERIC ALLENDER, AND DAVID A. MIX BARRINGTON: Uniform constant-depth threshold circuits for division and iterated multiplication. *J. Comput. System Sci.*, 65(4):695–716, 2002. With 2014 update in JCSS, Vol. 80(2), 496–497. [doi:10.1016/j.jcss.2013.09.002] 28

[22] ROGER A. HORN AND CHARLES R. JOHNSON: *Matrix Analysis*. Cambridge Univ. Press, 1990. [doi:10.1017/CBO9781139020411] 20

[23] RUSSELL IMPAGLIAZZO, NOAM NISAN, AND AVI WIGDERSON: Pseudorandomness for network algorithms. In *Proc. 26th STOC*, pp. 356–364. ACM Press, 1994. [doi:10.1145/195058.195190] 3

[24] GORAV JINDAL, PAVEL KOLEV, RICHARD PENG, AND SAURABH SAWLANI: Density independent algorithms for sparsifying k-step random walks. In *Proc. 20th Internat. Workshop on Approximation Algorithms for Combinat. Opt. Probl. (APPROX'17)*, pp. 14:1–17. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. [doi:10.4230/LIPIcs.APPROX-RANDOM.2017.14] 4

[25] RASMUS KYNG, JAKUB PACHOCKI, RICHARD PENG, AND SUSHANT SACHDEVA: A framework for analyzing resparsification algorithms. In *Proc. 28th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA'17)*. SIAM, 2017. [doi:10.1137/1.9781611974782.132, arXiv:1611.06940] 29, 30

[26] YIN TAT LEE, RICHARD PENG, AND DANIEL A. SPIELMAN: Sparsified Cholesky solvers for SDD linear systems. 2015. [arXiv:1506.08204] 4

[27] PO-LING LOH AND MARTIN J. WAINWRIGHT: Structure estimation for discrete graphical models: Generalized covariance matrices and their inverses. In *Proc. 25th Adv. Neural Info. Proc. Sys. (NIPS'12)*, pp. 2087–2095. Curran Assoc., 2012. NIPS. 27

[28] GRIGORY A. MARGULIS: Explicit constructions of expanders. *Problemy Peredachi Informatsii*, 9(4):71–80, 1973. MathNet.ru. 4

[29] GARY L. MILLER AND RICHARD PENG: Approximate maximum flow on separable undirected graphs. In *Proc. 24th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA'13)*, pp. 1151–1170. SIAM, 2013. [doi:10.1137/1.9781611973105.83] 17, 18

[30] JACK MURTAGH, OMER REINGOLD, AARON SIDFORD, AND SALIL P. VADHAN: Derandomization beyond connectivity: Undirected Laplacian systems in nearly logarithmic space. In *Proc. 58th FOCS*, pp. 801–812. IEEE Comp. Soc., 2017. [doi:10.1109/FOCS.2017.79] 3, 4, 6, 11, 13, 25

[31] JACK MURTAGH, OMER REINGOLD, AARON SIDFORD, AND SALIL P. VADHAN: Deterministic approximation of random walks in small space. In *Proc. 23rd Internat. Conf. on Randomization and Computation (RANDOM'19)*, pp. 42:1–22. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. [doi:10.4230/LIPIcs.APPROX-RANDOM.2019.42] 1, 5

[32] JOSEPH NAOR AND MONI NAOR: Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993. [doi:10.1137/0222053] 4

[33] RICHARD PENG AND DANIEL A. SPIELMAN: An efficient parallel solver for SDD linear systems. In *Proc. 46th STOC*, pp. 333–342. ACM Press, 2014. [doi:10.1145/2591796.2591832] 3, 28, 29

[34] OMER REINGOLD: Undirected connectivity in log-space. *J. ACM*, 55(4):17:1–24, 2008. [doi:10.1145/1391289.1391291] 2, 4, 25

[35] OMER REINGOLD, SALIL VADHAN, AND AVI WIGDERSON: Entropy waves, the zig-zag graph product, and new constant-degree expanders. *Ann. Math.*, 155(1):157–187, 2002. [doi:10.2307/3062153] 4, 9, 13

[36] EYAL ROZENMAN AND SALIL VADHAN: Derandomized squaring of graphs. In *Proc. 9th Internat. Workshop on Randomization and Computation (RANDOM'05)*, pp. 436–447. Springer, 2005. [doi:10.1007/11538462_37] 3, 4, 9, 10, 11, 13, 25

[37] MICHAEL SAKS AND SHIYU ZHOU: $\mathrm{BP_HSPACE}(S) \subseteq \mathrm{DSPACE}(S^{3/2})$. *J. Comput. System Sci.*, 58(2):376–403, 1999. Preliminary version in FOCS'95. [doi:10.1006/jcss.1998.1616] 2

[38] DANIEL A. SPIELMAN AND SHANG-HUA TENG: Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proc. 36th STOC*, pp. 81–90. ACM Press, 2004. [doi:10.1145/1007352.1007372] 6

[39] DANIEL A. SPIELMAN AND SHANG-HUA TENG: Spectral sparsification of graphs. *SIAM J. Comput.*, 40(4):981–1025, 2011. [doi:10.1137/08074489X] 3

[40] THOMAS STEINKE: Pseudorandomness for permutation branching programs without the group theory. *Electron. Colloq. Comput. Complexity*, TR12-083, 2012. [ECCC] 3

[41] R. MICHAEL TANNER: Explicit concentrators from generalized N-gons. *SIAM J. Alg. Discr. Methods*, 5(3):287–293, 1984. [doi:10.1137/0605030] 6

AUTHORS

Jack Murtagh
School of Engineering & Applied Sciences
Harvard University
Cambridge, MA USA
murtagh.jack@g.harvard.edu
https://scholar.harvard.edu/jmurtagh/home

Omer Reingold
Professor
Computer Science Department
Stanford University
Stanford, CA USA
reingold@stanford.edu
https://engineering.stanford.edu/people/omer-reingold

Aaron Sidford
Assistant professor
Management Science & Engineering
and Computer Science
Stanford University
Stanford, CA USA
sidford@stanford.edu
http://www.aaronsidford.com

Salil Vadhan
Professor
School of Engineering & Applied Sciences
Harvard University
Cambridge, MA USA
salil_vadhan@harvard.edu
http://salil.seas.harvard.edu

ABOUT THE AUTHORS

JACK MURTAGH got his Ph. D. in May 2020 from Harvard University where he was advised by Salil Vadhan. As an undergraduate, he studied math at Tufts University. Jack wrote his dissertation on spectral graph-theoretic methods for derandomizing space-bounded computation.

OMER REINGOLD is the Rajeev Motwani Professor in Computer Science at Stanford University. His research is in the foundations of computer science and most notably in computational complexity, cryptography, and the societal impact of computation. Omer completed his Ph. D. at the Weizmann Institute under the supervision of Moni Naor.

AARON SIDFORD is an assistant professor in Management Science and Engineering and in Computer Science at Stanford University. He received his Ph. D. from the Electrical Engineering and Computer Science Department at the Massachusetts Institute of Technology where he was advised by Jonathan Kelner. Aaron's research interests lie broadly in optimization, the theory of computation, and the design and analysis of algorithms. He is particularly interested in work at the intersection of continuous optimization, graph theory, numerical linear algebra, and data structures.

SALIL VADHAN is the Vicky Joseph Professor of Computer Science and Applied Mathematics at the Harvard John A. Paulson School of Engineering & Applied Sciences. He received his Ph. D. under the supervision of Shafi Goldwasser at MIT in 1999; the title of his dissertation was "A Study of Statistical Zero-Knowledge Proofs." Other research interests include the theory of pseudorandomness the theory and practice of data privacy. He enjoys spending leisure time with his wife and two daughters, as well as learning to surf in the cold waters of New England.