Derandomization Beyond Connectivity:
Undirected Laplacian Systems in Nearly Logarithmic Space

Jack Murtagh*, Omer Reingold†, Aaron Sidford‡, and Salil Vadhan*

*School of Engineering & Applied Sciences, Harvard University, Cambridge, MA USA
Email: jmurtagh@g.harvard.edu, salil@seas.harvard.edu
†Computer Science Department, Stanford University, Stanford, CA USA
Email: reingold@stanford.edu
‡Management Science & Engineering, Stanford University, Stanford, CA USA
Email: sidford@stanford.edu

Abstract—We give a deterministic $\tilde{O}(\log n)$-space algorithm for approximately solving linear systems given by Laplacians of undirected graphs, and consequently also approximating hitting times, commute times, and escape probabilities for undirected graphs. Previously, such systems were known to be solvable by randomized algorithms using $O(\log n)$ space (Doron, Le Gall, and Ta-Shma, 2017) and hence by deterministic algorithms using $O(\log^{3/2} n)$ space (Saks and Zhou, FOCS 1995 and JCSS 1999).

Our algorithm combines ideas from time-efficient Laplacian solvers (Spielman and Teng, STOC '04; Peng and Spielman, STOC '14) with ideas used to show that UNDIRECTED S-T CONNECTIVITY is in deterministic logspace (Reingold, STOC '05 and JACM '08; Rozenman and Vadhan, RANDOM '05).

Keywords-space complexity; derandomization; expander graphs; spectral sparsification; random walks; linear systems

I. INTRODUCTION

A. The RL vs. L Problem

One of the central problems in computational complexity is to understand the power of randomness for efficient computation. There is strong evidence that randomness does not provide a substantial savings for standard algorithmic problems, where the input is static and the algorithm has time to access it in entirety. Indeed, under widely believed complexity assumptions (e.g. that SAT has circuit complexity $2^{\Omega(n)}$), it is known that every randomized algorithm can be made deterministic with only a polynomial slowdown ($\mathbf{BPP} = \mathbf{P}$, $\mathbf{RP} = \mathbf{P}$) and a constant-factor increase in space ($\mathbf{BPL} = \mathbf{L}$, $\mathbf{RL} = \mathbf{L}$) [1], [2].

A major challenge is to prove such derandomization results unconditionally, without relying on unproven complexity assumptions. In the time-bounded case, it is known that proving that $\mathbf{RP} = \mathbf{P}$ requires progress on circuit lower bounds [3], [4]. But for the space-bounded case, there are no such barriers, and we can hope for an unconditional proof that $\mathbf{RL} = \mathbf{L}$. Indeed, Nisan [5] gave an unconditional construction of a pseudorandom generator for space-bounded computation with seed length $O(\log^2 n)$, which was used by Saks and Zhou [6] to prove that

$\mathbf{RL} \subseteq \mathbf{SPACE}(\log^{3/2} n) \stackrel{\text{def}}{=} \mathbf{L}^{3/2}$. Unfortunately, despite much effort over the past two decades, this remains the best known upper bound on the deterministic space complexity of $\mathbf{RL}$.

For many years, the most prominent example of a problem in $\mathbf{RL}$ not known to be in $\mathbf{L}$ was UNDIRECTED S-T CONNECTIVITY, which can be solved in randomized logarithmic space by performing a polynomial-length random walk from the start vertex s and accepting if the destination vertex t is ever visited [7]. In 2005, Reingold [8] gave a deterministic logspace algorithm for this problem. Since UNDIRECTED S-T CONNECTIVITY is complete for $\mathbf{SL}$ (symmetric logspace) [9], this also implies deterministic logpsace algorithms for several other natural problems, such as BIPARTITENESS [10], [11].

Reingold's algorithm provided hope for renewed progress on the general $\mathbf{RL}$ vs. $\mathbf{L}$ problem. Indeed, it was shown in [12] that solving S-T connectivity on *directed* graphs where the random walk is promised to have polynomial mixing time (and where t has noticeable stationary probability) is complete for $\mathbf{RL}$ (generalized to promise problems). While [12] was also able to generalize Reingold's methods to Eulerian directed graphs, the efforts to handle all of $\mathbf{RL}$ with this approach stalled. Thus, researchers turned back to constructing pseudorandom generators for more restricted models of space-bounded computation, such as various types of constant-width branching programs [13], [14], [15], [16], [17], [18], [19], [20], [21].

B. Our Work

In this paper, we restart the effort to obtain deterministic logspace algorithms for increasingly rich graph-theoretic problems beyond those known to be in $\mathbf{SL}$. In particular, we provide a nearly logarithmic space algorithm for approximately solving UNDIRECTED LAPLACIAN SYSTEMS, which also implies nearly logarithmic-space algorithms for approximating hitting times, commute times, and escape probabilities for random walks on undirected graphs.

Our algorithms are obtained by combining the techniques from recent nearly linear-time Laplacian solvers [22], [23] with methods related to Reingold's algorithm [24]. The body of work of time-efficient Laplacian solvers has recently been extended to Eulerian directed graphs [25], which in turn was used to obtain algorithms to approximate stationary probabilities for arbitrary directed graphs with polynomial mixing time through recent reductions [26].

This raises the tantalizing possibility of extending our nearly logarithmic-space Laplacian solvers in a similar way to prove that $\mathbf{RL} \subseteq \tilde{\mathbf{L}} \stackrel{\text{def}}{=} \mathbf{SPACE}(\tilde{O}(\log n))$, since approximating stationary probabilities on digraphs with polynomial mixing time suffices to solve all of $\mathbf{RL}$ [12], [27], and Reingold's algorithm has been extended to Eulerian digraphs [12], [24].

C. Laplacian system solvers

Given an undirected multigraph G with adjacency matrix A and diagonal degree matrix D, the *Laplacian* of G is the matrix $L = D - A$. Solving systems of equations $Lx = b$ in the Laplacians of graphs (and in general symmetric diagonally dominant systems) arises in many applications throughout computer science. In a seminal paper, Spielman and Teng gave a nearly linear time algorithm for solving such linear systems [22], which was improved in a series of follow up works [28], [29], [30], [23], [31] including recent extensions to directed graphs [26], [25]. These methods have sparked a large body of work using Laplacian solvers as an algorithmic primitive for problems such as max flow [32], [33], randomly sampling spanning trees [34], sparsest cut [35], graph sparsification [36], as well as problems in computer vision [37].

Recent works have started to study the space complexity of solving Laplacian systems. Ta-Shma [38] gave a *quantum* logspace algorithm for approximately solving general linear systems that are suitably well-conditioned. Doron, Le Gall, and Ta-Shma [39] showed that there is a *randomized* logspace algorithm for approximately solving Laplacian systems on digraphs with polynomial mixing time.

D. Main Result

We give a nearly logarithmic-space algorithm for approximately solving undirected Laplacian systems:

Theorem I.1. *There is a deterministic algorithm that, given an undirected multigraph $G = (V, E)$ specified as a list of edges, a vector $b \in \mathbb{Q}^{|V|}$ in the image of G's Laplacian L, and an approximation parameter $\epsilon > 0$, finds a vector $x \in \mathbb{Q}^{|V|}$, such that*

$$\|x - x^*\| \leq \epsilon \cdot \|x^*\|$$

for some x^* such that $Lx^* = b$, in space

$$O\left(\log n \cdot \log \log(n/\epsilon)\right),$$

where n is the bitlength of the input (G, b, ϵ).

In particular, for $\epsilon = 1/\text{poly}(n)$, the algorithm uses space $\tilde{O}(\log n)$. Note that since the algorithm applies to multigraphs (represented as edge lists), we can handle polynomially large integer edge weights. Known reductions [26] imply the same space bounds for estimating hitting times, commute times, and escape probabilities for random walks on undirected graphs. (See Section VIII.)

E. Techniques

The starting point for our algorithm is the undirected Laplacian solver of Peng and Spielman [23], which is a randomized algorithm that uses polylogarithmic parallel time and a nearly linear amount of work. (Interestingly, concurrently and independently of [8], Trifonov [40] gave a $\tilde{O}(\log n)$-space algorithm for UNDIRECTED S-T CONNECTIVITY by importing techniques from parallel algorithms, like we do.) It implicitly[1] computes an approximate *pseudoinverse* L^+ of a graph Laplacian (formally defined in Section II-B), which is equivalent to approximately solving Laplacian systems. Here we will sketch their algorithm and how we obtain a space-efficient analogue of it.

By using the deterministic logspace algorithm for UNDIRECTED S-T CONNECTIVITY [8], we may assume without loss of generality that our input graph G is connected (else we find the connected components and work on each separately). By adding self-loops (which does not change the Laplacian), we may also assume that G is regular and nonbipartite. For notational convenience, here and through most of the paper we will work with the *normalized Laplacian*, which in the case that G is regular,[2] equals $L = L(G) = I - M$, where $M = M(G)$ is the $n \times n$ transition matrix for the random walk on G and now we are using n for the number of vertices in G. Since the uniform distribution is stationary for the random walk on a regular graph, the all-ones vector $\vec{1}$ is in the kernel of L. Because G is connected, there is no other stationary distribution and L is of rank $n - 1$. Thus computing L^+ amounts to inverting L on the space orthogonal to $\vec{1}$.

The Peng–Spielman algorithm is based on the following identity:

$$(I - M)^+ = \frac{1}{2} \cdot \left(I - J + (I + M)(I - M^2)^+(I + M)\right),$$

where J is the matrix with all entries $1/n$. That is,

$$L(G)^+ = \frac{1}{2} \cdot \left(I - J + (I + M(G))L(G^2)^+(I + M(G))\right), \tag{1}$$

---

[1]Nearly linear-time algorithms for Laplacian system solvers do not have enough time to write done a full approximate pseudoinverse, which may be dense; instead they compute the result of applying the approximate matrix to a given vector.

[2]In an irregular graph, the normalized Laplacian is defined to be $I - D^{-1/2}AD^{-1/2}$, where D is the diagonal matrix of vertex degrees. When G is d-regular, we have $D = dI$, so $D^{-1/2}AD^{-1/2} = A/d = M$.

where G^2 is the multigraph whose edges correspond to walks of length 2 in G.

This gives rise to a natural algorithm for computing $L(G)^+$, by recursively computing $L(G^2)^+$ and applying Equation (1). After squaring the graph $k = O(\log n)$ times, we are considering all walks of length 2^k, which is beyond the $O(n^2)$ mixing time of regular graphs, and hence $M(G^{2^k})$ is approximately J, and we can easily approximate the pseudoinverse as $L(G^{2^k})^+ \approx (I - J)^+ = I - J$, the Laplacian of the complete graph with self loops on every vertex.

However, each time we square the graph, the degree also squares, which is too costly in terms of computational complexity. Thus, to obtain nearly linear time, Peng and Spielman [23] sparsify G^2 at each step of recursion. Specifically, they carefully combine elements of randomized sparsification procedures from [41], [42], [43] that retains $\tilde{O}(n)/\epsilon^2$ edges from G^2 and provides a spectral ϵ-approximation G' to G^2 in the sense that for every vector v, we have $v^T L(G')v = (1 \pm \epsilon)v^T L(G^2)v$. The recursion of Equation (1) behaves nicely with respect to spectral approximation, so that if we replace G^2 with such an ϵ-approximation G' at each step, we obtain a $O(k\epsilon)$-approximation over the $k = O(\log n)$ levels of recursion provided $\epsilon \leq 1/O(k)$. Thus, we can take $\epsilon = 1/O(\log n)$ and obtain a good spectral approximation overall, using graphs with $\tilde{O}(n)$ edges throughout and thereby maintaining a nearly linear amount of work.

There are two issues with trying to obtain a deterministic $\tilde{O}(\log n)$-space algorithm from this approach. First, we cannot perform random sampling to sparsify. Second, even if we derandomize the sampling step in deterministic logarithmic space, a direct recursive implementation would cost $O(\log n)$ space for each of the $k = O(\log n)$ levels of recursion, for a space bound of $O(\log^2 n)$.

For the first issue, we show that the sparsification can be done deterministically using the *derandomized square* of Rozenman and Vadhan [24], which was developed in order to give a simpler proof that UNDIRECTED S-T CONNECTIVITY is in **L**. If G is a d-regular graph, a derandomized square of G is obtained by using an expander H on d vertices to select a subset of the walks of length 2 in G. If H has degree $c \ll d$, then the corresponding derandomized square of G will have degree $d \cdot c \ll d^2$. In [24], it was shown that the derandomized square improves expansion (as measured by spectral gap) nearly as well actual squaring, to within a factor of $1 - \epsilon$, where ϵ is the second largest eigenvalue of H in absolute value. Here we show that derandomized squaring actually provides a spectral $O(\epsilon)$-approximation to G^2. Consequently, we can use derandomized squaring with expanders of second eigenvalue $1/O(\log n)$ and hence degree $c = \text{polylog}(n)$ in the Peng–Spielman algorithm.

Next, to obtain a space-efficient implementation, we consider what happens when we repeatedly apply Equation (1),

but with true squaring replaced by derandomized squaring. We obtain a sequence of graphs $G_0, G_1, \ldots, G_k$, where $G_0 = G$ and G_i is the derandomized square of G_{i-1}, a graph of degree $d \cdot c^i$. We recursively compute spectral approximations of the pseudoinverses $L(G_i)^+$ as follows:

$$\widetilde{L(G_{i-1})}^+ = \frac{1}{2} \cdot \left(I - J + (I + M(G_{i-1}))\widetilde{L(G_i)}^+(I + M(G_{i-1})) \right).$$
(2)

Opening up all $k = O(\log n)$ levels of this recursion, there is an explicit *quadratic* matrix polynomial p where each of the (noncommuting) variables appears at most twice in each term, such that

$$\widetilde{L(G_0)} = p(I + M(G_0), I + M(G_1), \ldots, I + M(G_k)).$$

Our task is to evaluate this polynomial in space $\tilde{O}(\log n)$. First, we use the fact, following [8], [24], that a careful implementation of k-fold derandomized squaring using explicit expanders of degree c can be evaluated in space $O(\log n + k \cdot \log c) = \tilde{O}(\log n)$. (A naive evaluation would take space $O(k \cdot \log n) = O(\log^2 n)$.) This means that we can construct each of the matrices $I + M(G_i)$ in space $\tilde{O}(\log n)$.[3] Next, we observe that each term in the multilinear polynomial multiplies at most $2k + 1$ of these matrices together, and hence can be computed recursively in space $O(\log k) \cdot O(\log n) = \tilde{O}(\log n)$. Since iterated addition is also in logspace, we can then sum to evaluate the entire matrix polynomial in space $\tilde{O}(\log n)$.

To obtain an arbitrarily good final approximation error $\epsilon > 0$, we could use expanders of degree $c = \text{poly}((\log n)/\epsilon)$ for our derandomized squaring, but this would yield a space complexity of at least $k \cdot \log n \geq \log(1/\epsilon) \cdot \log n$. To obtain the doubly-logarithmic dependence on $1/\epsilon$ claimed in Theorem I.1, we follow the same approach as [23], computing a constant factor spectral approximation as above, and then using "Richardson iterations" at the end to improve the approximation factor. Interestingly, this doubly-logarithmic dependence on ϵ is even better than what is achieved by the randomized algorithm of [39], which uses space $O(\log(n/\epsilon))$.

II. PRELIMINARIES

A. Graph Laplacians

Let $G = (V, E)$ be an undirected multigraph on n vertices. The *adjacency matrix* A is the $n \times n$ matrix such that entry A_{ij} contains the number of edges from vertex j to vertex i in G. The *degree matrix* D is the diagonal $n \times n$

---

[3]We follow the usual model of space-bounded computation, where algorithms can have output (to write-only memory) that is larger than their space bound. We review the standard composition lemma for such algorithms in Section VI-A.

matrix such that D_{ii} is the degree of vertex i in G. The *Laplacian* of G is defined to be

$$L = D - A.$$

L is a symmetric, positive semidefinite matrix with non-positive off-diagonal entries and diagonal entries $L_{ii} = \sum_{j \neq i} |L_{ij}|$ equaling the sum of the absolute values of the off-diagonal entries in that row. Each row of a Laplacian sums to 0 so the all-ones vector is in the kernel of every Laplacian.

We will also work with the normalized Laplacian of regular graphs $LD^{-1} = I - M$ where $M = AD^{-1}$ is the transition matrix of the random walk on the graph G.

Definition II.1. Let G be a regular undirected graph with transition matrix M. Then we define

$$\lambda(G) = \max_{\substack{v \perp \vec{1} \\ v \neq 0}} \frac{\|Mv\|}{\|v\|}$$

$$= \text{2nd largest eigenvalue of } M \text{ in absolute value.}$$

The *spectral gap* of G is $\gamma(G) = 1 - \lambda(G)$

The spectral gap of a multigraph is a good measure of its expansion properties. It is shown in [24] that if G is a d-regular multigraph on n vertices with a self loop on every vertex then $\lambda(G) \leq 1 - 1/2d^2n^2$.

Throughout the paper we let J denote the $n \times n$ matrix with $1/n$ in every entry. So $I - J$ is the normalized Laplacian of the complete graph with a self loop on every vertex. We use $\vec{1}$ to denote the all ones vector.

B. Moore-Penrose pseudoinverse

Since a normalized Laplacian $L = I - M$ is not invertible, we will consider the Moore-Penrose pseudoinverse of L.

Definition II.2. Let L be a real-valued matrix. The *Moore-Penrose pseudoinverse* L^+ of L is the unique matrix satisfying the following:
1) $LL^+L = L$
2) $L^+LL^+ = L^+$
3) $L^+L = (L^+L)^{\intercal}$
4) $LL^+ = (LL^+)^{\intercal}$.

Fact II.3. *For a real-valued matrix L, L^+ has the following properties:*
1) $\ker(L^+) = \ker(L^{\intercal})$
2) $\operatorname{im}(L^+) = \operatorname{im}(L^{\intercal})$
3) *If c is a nonzero scalar then $(cL)^+ = c^{-1}L^+$*
4) L^+ *is real-valued*
5) *If L is symmetric with eigenvalues $\lambda_1, \ldots, \lambda_n$, then L^+ has the same eigenvectors as L and eigenvalues $\lambda_1^+, \ldots, \lambda_n^+$ where*

$$\lambda_i^+ = \begin{cases} \dfrac{1}{\lambda_i} & \text{if } \lambda_i \neq 0 \\ 0 & \text{if } \lambda_i = 0. \end{cases}$$

Next we show that solving a linear system in L can be reduced to computing the pseudoinverse of L and applying it to a vector.

Proposition II.4. *If $Lx = b$ has a solution, then the vector L^+b is a solution.*

Proof: Let x^* be a solution. Multiplying both sides of the equation $Lx^* = b$ by LL^+ gives

$$LL^+Lx^* = LL^+b$$
$$\implies Lx^* = LL^+b$$
$$\implies b = LL^+b$$

The final equality shows that L^+b is a solution to the system. ∎

C. Spectral approximation

We want to approximate a solution to $Lx = b$. Our algorithm works by computing an approximation $\tilde{L}^+$ to the matrix L^+ and then outputting $\tilde{L}^+b$ as an approximate solution to $Lx = b$. We use the notion of spectral approximation of matrices first introduced in [41].

Definition II.5. Let X, Y be symmetric $n \times n$ real matrices. We say that $X \approx_\epsilon Y$ if

$$e^{-\epsilon} \cdot X \preceq Y \preceq e^\epsilon \cdot X$$

where for any two matrices A, B we write $A \preceq B$ if $B - A$ is positive semidefinite.

Below are some additional useful facts about spectral approximation that we use throughout the paper.

Proposition II.6 ([23])**.** *If X, Y, W, Z are positive semidefinite $n \times n$ matrices then*
1) *If $X \approx_\epsilon Y$ then $Y \approx_\epsilon X$*
2) *If $X \approx_\epsilon Y$ and $c \geq 0$ then $cX \approx_\epsilon cY$*
3) *If $X \approx_\epsilon Y$ then $X + W \approx_\epsilon Y + W$*
4) *If $X \approx_\epsilon Y$ and $W \approx_\epsilon Z$ then $X + W \approx_\epsilon Y + Z$*
5) *If $X \approx_{\epsilon_1} Y$ and $Y \approx_{\epsilon_2} Z$ then $X \approx_{\epsilon_1 + \epsilon_2} Z$*
6) *If $X \approx_\epsilon Y$ and M is any matrix then $M^{\intercal}XM \approx_\epsilon M^{\intercal}YM$*
7) *If $X \approx_\epsilon Y$ and X and Y have the same kernel, then $X^+ \approx_\epsilon Y^+$*
8) *If $X \approx_\epsilon Y$ then $I \otimes X \approx_\epsilon I \otimes Y$ where I is the identity matrix (of any dimension) and $\otimes$ denotes the tensor product.*

The proof of Proposition II.6 can be found in the full version of this paper.

III. MAIN THEOREM AND APPROACH

Our Laplacian solver works by approximating L^+. The main technical result is:

Theorem III.1. *Given an undirected, connected multigraph G with Laplacian $L = D - A$ and $\epsilon > 0$, there is a deterministic algorithm that computes $\tilde{L}^+$ such that $\tilde{L}^+ \approx_\epsilon L^+$ that uses space $O(\log n \cdot \log\log(n/\epsilon))$ where n is the bitlength of the input.*

Approximating solutions to Laplacian systems follows as a corollary and is discussed in Section VIII. To prove Theorem III.1 we first reduce the problem to the task of approximating the pseudoinverse of the normalized Laplacian of a regular, aperiodic, multigraph with degree a power of 2. Let $L = D - A$ be the Laplacian of an undirected multigraph G. We can make G f-regular with f a power of 2 and aperiodic by adding an appropriate number of self loops to every vertex. Let E be the diagonal matrix of self loops added to each vertex in G. Then $L = D + E - A - E$ is the Laplacian of the regular, aperiodic multigraph (self loops do not change the unnormalized Laplacian) and $L(D + E)^{-1} = L/f$ is the normalized Laplacian. Recalling that $(L/f)^+ = f \cdot L^+$ completes the reduction.

Our algorithm for computing the pseudoinverse of the normalized Laplacian of a regular, aperiodic multigraph is based on the Laplacian solver of Peng and Spielman [23]. It works by using the following identity:

Proposition III.2. *If $L = I - M$ is the normalized Laplacian of an undirected, connected, regular, aperiodic, multigraph G on n vertices then*
$$L^+ = \frac{1}{2}\left(I - J + (I + M)(I - M^2)^+(I + M)\right)$$

This identity is adapted from the one used in [23]. A proof of its correctness can be found in the full version of this paper.

Recall that squaring the transition matrix M of a regular multigraph G yields the transition matrix of G^2, which is defined to be the graph on the same vertex set as G whose edges correspond to walks of length 2 in G. So the identity from Proposition III.2 reduces the problem of computing the pseudoinverse of the normalized Laplacian of G to computing the pseudoinverse of the normalized Laplacian of G^2 (plus some additional matrix products). Recursively applying the identity k times results in a matrix polynomial involving the term $(I - M^{2^k})^+$. Since G is connected and aperiodic, as k grows, the term $(I - M^{2^k})^+$ approaches $(I - J)^+ = I - J$. Undirected multigraphs have polynomial mixing time, so setting k to $O(\log n)$ and then replacing $(I - M^{2^{k+1}})^+$ in the final term of the expansion with $(I - J)$ should give a good approximation to L^+ without explicitly computing any pseudoinverses.

Using the identity directly to approximate L^+ in $\tilde{O}(\log n)$ space is infeasible because raising a matrix to the 2^kth power by repeated squaring takes space $\Theta(k \cdot \log n)$, which is $\Theta(\log^2 n)$ for $k = \Theta(\log n)$.

To save space we use the derandomized square introduced in [24] in place of true matrix squaring. The derandomized

square improves the connectivity of a graph almost as much as true squaring does, but it does not square the degree and thereby avoids the space blow-up when iterated.

IV. Approximation of the Pseudoinverse

In this section we show how to approximate the pseudoinverse of a Laplacian using a method from Peng and Spielman [23]. The Peng Spielman solver was originally written to approximately solve symmetric diagonally dominant systems, which in turn can be used to approximate solutions to Laplacian systems. We adapt their algorithm to compute the pseudoinverse of a graph Laplacian directly. The approximation proceeds in two steps. The first achieves a constant spectral approximation to the pseudoinverse of a Laplacian matrix. Then the constant approximation is boosted to an ϵ approximation through rounds of Richardson iteration.

Theorem IV.1 (Adapted from [23]). *Let $\epsilon_0, \ldots, \epsilon_k \geq 0$ and $\epsilon = \sum_0^k \epsilon_i$. Let $M_0, \ldots, M_k$ be symmetric matrices such that $L_i = I - M_i$ are positive semidefinite for all $i \in \{0, \ldots, k\}$ and $L_i \approx_{\epsilon_{i-1}} I - M_{i-1}^2$ for all $1 \leq i \leq k$ and $L_k \approx_{\epsilon_k} I - J$. Then*

$$L_0^+ \approx_\epsilon \frac{1}{2}(I - J) + \left(\sum_{i=0}^{k-1} \frac{1}{2^{i+2}} W_i\right) + \frac{1}{2^{k+1}} W_k$$

where for all $i \in [k]$

$$W_i = (I + M_0) \cdot \ldots \cdot (I + M_i)(I - J)(I + M_i) \cdot \ldots \cdot (I + M_0)$$

The proof of this theorem is adapted from [23] and can be found in the full version of this paper.

Our algorithm works by using Theorem IV.1 to compute a constant approximation to L^+ and then boosting the approximation to an arbitrary ϵ-approximation. Our main tool for this is the following lemma which shows how an approximate pseudoinverse can be improved to a high quality approximate pseudoinverse. This is essentially a symmetric version of a well known technique in linear system solving known as preconditioned Richardson iteration. It is a slight modification of Lemma 31 from [44].

Lemma IV.2. *Let $\tilde{L}^+ \approx_\alpha L^+$ for $\alpha < 1/2$. Then*

$$L_k = e^{-\alpha} \cdot \sum_{i=0}^k L \cdot (I - e^{-\alpha} \cdot \tilde{L}^+ \cdot L)^i$$

is a matrix satisfying $L_k \approx_{O((2\alpha)^k)} L^+$

The proof of Lemma IV.2 can be found in the full version of this paper.

V. Derandomized Squaring

To approximate the pseudoinverse of a graph Laplacian we will use Theorem IV.1 to get a constant approximation and then use Lemma IV.2 to boost the approximation. In

order to achieve a space efficient and deterministic implementation of Theorem IV.1, we will replace every instance of matrix squaring with the derandomized square of Rozenman and Vadhan [24]. Before defining the derandomized square we define two-way labelings and rotation maps.

Definition V.1. [45] A *two-way labeling* of a d-regular undirected multigraph G is a labeling of the edges in G such that

1) Every edge (u, v) has two labels in $[d]$, one as an edge incident to u and one as an edge incident to v.
2) For every vertex v the labels of the edges incident to v are distinct.

In [24], two-way labelings are referred to as *undirected two-way labelings*. In a two-way labeling, each vertex has its own labeling from 1 to d for the d edges incident to it. Since every edge is incident to two vertices, each edge receives two labels, which may or may not be the same. It is convenient to specify a multigraph with a two-way labeling by a rotation map:

Definition V.2. [45] Let G be a d-regular multigraph on n vertices with a two-way labeling. The *rotation map* $\text{Rot}_G \colon [n] \times [d] \to [n] \times [d]$ is defined as follows: $\text{Rot}_G(v, i) = (w, j)$ if the ith edge incident to vertex v leads to vertex w and this edge is the jth edge incident to w.

Note that Rot_G is its own inverse, and that any function $\text{Rot} \colon [n] \times [d] \to [n] \times [d]$ that is its own inverse defines a d-regular multigraph on n vertices with a two-way labeling. Recall that the edges in G^2 correspond to all of the walks of length 2 in G. This is equivalent to placing a d-clique with self loops on every vertex on the neighbor set of every vertex in G. The derandomized square picks out a subset of the walks of length 2 by placing a small degree expander on the neighbor set of every vertex rather than a clique.

Definition V.3 ([24])**.** Let G be a d-regular multigraph on n vertices with a two-way labeling, let H be a c-regular multigraph on d vertices with a two-way labeling. The *derandomized square* $G \text{ⓢ} H$ is a $(d \cdot c)$-regular graph on n vertices with rotation map $\text{Rot}_{G \text{ⓢ} H}$ defined as follows: For $v_0 \in [n], i_0 \in [d]$, and $j_0 \in [c]$, we compute $\text{Rot}_{G \text{ⓢ} H}(v_0, (i_0, j_0))$ as

1) Let $(v_1, i_1) = \text{Rot}_G(v_0, i_0)$
2) Let $(i_2, j_1) = \text{Rot}_H(i_1, j_0)$
3) Let $(v_2, i_3) = \text{Rot}_G(v_1, i_2)$
4) Output $(v_2, (i_3, j_1))$

It can be verified that $\text{Rot}_{G \text{ⓢ} H}$ is its own inverse and hence this indeed defines a $(d \cdot c)$-regular multigraph on n vertices. The main idea behind the derandomized square is that it improves the connectivity of the graph (as measured by the second eigenvalue) without squaring the degree.

Theorem V.4 ([24])**.** *Let G be a d-regular undirected multigraph with a two-way labeling and $\lambda(G) = \lambda$ and H be a c-regular graph with a two-way labeling and $\lambda(H) = \mu$. Then $G \text{ⓢ} H$ is a $d \cdot c$-regular undirected multigraph and*

$$\lambda(G \text{ⓢ} H) \leq 1 - (1 - \lambda^2) \cdot (1 - \mu) \leq \lambda^2 + \mu$$

The output of the derandomized square is an undirected multigraph with a two-way labeling. So the operation can be repeated on the output. In our algorithm for computing an approximate pseudoinverse, we use the Peng Spielman identity but replace every instance of a squared transition matrix with the derandomized square. To prove that this approach indeed yields an approximation to the pseudoinverse, we want to apply Theorem IV.1. For this we need to prove two properties of the derandomized square: that the derandomized square is a good spectral approximation of the true square and that iterating the derandomized square yields an approximation of the complete graph. The latter property follows from the corollary and lemma below.

Corollary V.5. *Let G_0 be a d-regular undirected multigraph on n vertices with a two-way labeling and $\lambda(G_0) = \lambda$ and $H_1, \ldots, H_k$ be c-regular graphs with two-way labelings where H_i has $d \cdot c^{i-1}$ vertices and $\lambda(H_i) \leq \mu$. For all $i \in [k]$ let $G_i = G_{i-1} \text{ⓢ} H_i$. If $\mu \leq 1/100$ and $k = \lceil 6 \log d^2 n^2 \rceil$ then $\lambda(G_k) \leq 1/3$.*

Proof: Let $\mu \leq 1/100$ and for all $i \in [k]$, let $\gamma_i = 1 - \lambda(G_i)$. If $\gamma_{i-1} < 2/3$ then

$$\begin{aligned}
\gamma_i &\geq (1 - \lambda(G_{i-1})^2) \cdot (1 - \mu) \\
&= (2\gamma_{i-1} - \gamma_{i-1}^2) \cdot (1 - \mu) \\
&\geq \frac{99}{100} \cdot \frac{4}{3}\gamma_{i-1} \\
&\geq \frac{5}{4}\gamma_{i-1}
\end{aligned}$$

It follows that until γ_i is driven above $2/3$ we have $\gamma_k \geq (5/4)^k \cdot \gamma_0$. Since $\gamma_0 \geq 1/2d^2 n^2$, setting $k = \lceil 6 \log d^2 n^2 \rceil$ will result in $\lambda(G_k) \leq 1/3$. ∎

Lemma V.6. *If $L = I - M$ is the normalized Laplacian of a d-regular, undirected, 1/2-lazy, multigraph G, and $\epsilon = \ln(1/(1 - \lambda))$ then $L \approx_\epsilon I - J$ if and only if $\lambda(G) \leq \lambda$.*

A proof of Lemma V.6 can be found in the full version of this paper.

We prove in Theorem V.7 that if H is an expander then the derandomized square of a multigraph G with H is a good spectral approximation to G^2 and thus can be used for the approximation in Theorem IV.1. The idea is that the square of a graph can be interpreted as putting d-clique K on the neighbors of each vertex v in G, and the derandomized square can be interpreted as putting a copy of H on each vertex. By Lemma V.6, $L(H)$ spectrally approximates $L(K)$, and we can use this to show that the Laplacian of $L(G \text{ⓢ} H)$ spectrally approximates $L(G^2)$.

Theorem V.7. *Let G be a d-regular, undirected, aperiodic multigraph on n vertices and H be a regular multigraph on d vertices with $\lambda(H) \leq \alpha$. Then*

$$L(G^2) \approx_\epsilon L(G \text{ⓢ} H)$$

for $\epsilon = \ln(1/(1-\alpha))$.

Proof: Following the proof from [24] that the derandomized square improves connectivity, we can write the transition matrix for the random walk on $G \text{ⓢ} H$ as $M = P\tilde{A}\tilde{B}\tilde{A}Q$, where each matrix corresponds to a step in the definition of the derandomized square:

- Q is an $n \cdot d \times n$ matrix that maps any n-vector v to $v \otimes u_d$ (where u_d denotes the d-coordinate uniform distribution). This corresponds to "lifting" a probability distribution over $[n]$ to one over $[n] \times [d]$ where the mass on each coordinate is divided uniformly over d coordinates in the distribution over $[n] \times [d]$. That is $Q_{(u,i),v} = 1/d$ if $u = v$ and 0 otherwise where the rows of Q are ordered $(1,1), (1,2), \ldots, (1,d), (2,1), \ldots, (2,d), \ldots (n,d)$.
- $\tilde{A}$ is the $n \cdot d \times n \cdot d$ permutation matrix corresponding to the rotation map for G. That is $A_{(u,i),(v,j)} = 1$ if $\text{Rot}_G(u,i) = (v,j)$ and 0 otherwise.
- $\tilde{B}$ is $I_n \otimes B$, where I_n is the $n \times n$ identity matrix and B is the transition matrix for H.
- P is the $n \times n \cdot d$ matrix that maps any $(n \cdot d)$-vector to an n-vector by summing all the entries corresponding to the same vertex in G. This corresponds to projecting distributions on $[n] \times [d]$ back down to a distribution over $[n]$. That is $P_{v,(u,i)} = 1$ if $u = v$ and 0 otherwise where the columns of P are ordered $(1,1), (1,2), \ldots, (1,d), (2,1), \ldots, (2,d), \ldots (n,d)$. Note that $P = d \cdot Q^\intercal$.

Let c be the degree of H and let K be the complete graph on d vertices with self loops on every vertex. Lemma V.6 says that $L_H \approx_\epsilon L_K$ where $L_H = I - B$ and $L_K = I - J$ are the normalized Laplacians of H and K, respectively. It follows that $I_n \otimes L_H = I_{n \cdot d} - \tilde{B}$ ϵ-approximates $I_n \otimes L_K = I_{n \cdot d} - I_n \otimes J$ by Proposition II.6 Part 8. $I_n \otimes L_H$ is the Laplacian for n disjoint copies of H and $I_n \otimes L_K$ is the Laplacian for n disjoint copies of K. Applying the matrices $P\tilde{A}$ and $\tilde{A}Q$ on the left and right place these copies on the neighborhoods of each vertex of G.

Note that $\tilde{A}$ is symmetric because Rot_G is an involution. In other words, for all $u,v \in [n]$ and $a,b \in [d]$, $\text{Rot}_G(u,a) = (v,b) \implies \text{Rot}_G(v,b) = (u,a)$. This also implies that $\tilde{A}^2 = I_{n \cdot d}$. Also note that $PQ = I_n$ and $QP = I_{n \cdot d} \otimes J$. Applying these observations along with Proposition II.6 part

6 we get

$$
\begin{aligned}
I_n - M &= P\tilde{A}I_{n \cdot d}\tilde{A}Q - P\tilde{A}\tilde{B}\tilde{A}Q \\
&= P\tilde{A}(I_n \otimes L_H)\tilde{A}Q \\
&= d \cdot Q^\intercal \tilde{A}^\intercal (I_n \otimes L_H)\tilde{A}Q \\
&\approx_\epsilon d \cdot Q^\intercal \tilde{A}^\intercal (I_n \otimes L_K)\tilde{A}Q \\
&= P\tilde{A}(I_n \otimes L_K)\tilde{A}Q \\
&= I_n - P\tilde{A}(I_n \otimes J)\tilde{A}Q \\
&= I_n - P\tilde{A}QP\tilde{A}Q \\
&= L(G^2)
\end{aligned}
$$

where the final line follows from observing that $P\tilde{A}Q$ equals the transition matrix of G. Thus the above shows that $L(G^2) \approx_\epsilon L(G \text{ⓢ} H)$ as desired. ∎

Note that if we want to compute a constant spectral approximation to L^+ then we need each $\epsilon_i = 1/O(\log n)$ so that the sum of $k = O(\log n)$ of them is a constant. This implies that we need $\alpha = 1/O(\log n)$ in Theorem V.7 which requires expanders of polylogarithmic degree. This will affect the space complexity of the algorithm as discussed in the next section.

VI. Space complexity

In this section we argue that given a Laplacian L, our algorithm produces an ϵ-approximation to L^+ using $O(\log n \cdot \log \log(n/\epsilon))$ space. First we prove that when L is the normalized Laplacian of a regular graph, we can produce a constant-approximation to L^+ using $O(\log n \cdot \log \log n)$ space and then boost the approximation quality to arbitrary ϵ accuracy using an additional $O(\log n \cdot \log \log(1/\epsilon))$ space thereby proving Theorem III.1.

A. Model of space bounded computation

We use a standard model of space bounded computation described here. The machine has a read-only input tape, a constant number of read/write work tapes, and a write-only output tape. We say the machine runs in space s if throughout the computation, it only uses s tape cells on the work tapes. The machine may write outputs to the output tape that are larger than s (in fact as large as $2^{O(s)}$) but the output tape is write-only. For the randomized analogue, we imagine the machine endowed with a coin-flipping box. At any point in the computation the machine may ask for a random bit but in order to remember any bit it receives, the bit must be recorded on a work tape.

We use the following proposition about the composition of space-bounded algorithms.

Proposition VI.1. *Let f_1 and f_2 be functions that can be computed in space $s_1(n), s_2(n) \geq \log n$, respectively, and f_2 has output of length $\ell_1(n)$ on inputs of size n. Then $f_2 \circ f_1$ can be computed in space*

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

B. Constant approximation of the pseudoinverse

In this section we show how to compute a constant approximation of the pseudoinverse of the normalized Laplacian of a regular multigraph in space $O(\log n \cdot \log \log n)$ where n is the bitlength of the input.

Proposition VI.2. *There is an algorithm that given an undirected, aperiodic, d-regular multigraph G with normalized Laplacian L, and d a power of 2, computes a matrix $\tilde{L}^+$ such that $\tilde{L}^+ \approx_\alpha L^+$ using space $O(\log n \cdot \log \log n)$ where $\alpha < 1/2$ and n is the bitlength of the input.*

To prove Proposition VI.2, we first prove a few lemmas. We will use the fact that large derandomized powers can be computed in small space, which we prove following [8], [24], [46]. First we note that neighbors in the sequence of expanders we use for the iterated derandomized square can be explicitly computed space-efficiently.

Lemma VI.3. *For every $t \in \mathbb{N}$ and $\mu > 0$, there is a graph $H_{t,\mu}$ with a two-way labeling such that:*

- *H has 2^t vertices and is c-regular for c being a power of 2 bounded by $\mathrm{poly}(t, 1/\mu)$.*
- *$\lambda(H) \leq \mu$*
- *Rot_H can be evaluated in linear space in its input length, i.e. space $O(t + \log c)$.*

Proof sketch.: The expanders can be obtained by letting $H_{t,\lambda}$ be the Cayley graph on vertex set $\mathbb{F}_2^t$ with neighbors specified by a λ-biased multiset $S \subseteq \mathbb{F}_2^t$ [47], [48], taking $\mathrm{Rot}_H(v, i) = (v + s_i, i)$, where s_i is the ith element of S. ∎

Now we argue that high derandomized powers can be computed space-efficiently.

Lemma VI.4. *Let G_0 be a d-regular, undirected multigraph on n vertices with a two-way labeling and $H_1, \ldots, H_k$ be c-regular undirected graphs with two-way labelings where for each $i \in [k]$, H_i has $d \cdot c^{i-1}$ vertices. For each $i \in [k]$ let*

$$G_i = G_{i-1} \textcircled{s} H_i.$$

Then given $v_0 \in [n], i_0 \in [d \cdot c^{i-1}], j_0 \in [c]$, $\mathrm{Rot}_{G_i}(v, (i_0, j_0))$ can be computed in space $O(\log(n \cdot d) + k \cdot \log c)$ with oracle queries to $\mathrm{Rot}_{H_1}, \ldots, \mathrm{Rot}_{H_k}$.

Proof: When c is subpolynomial we are reasoning about sublogarithmic space complexity, which can depend on the model. We will use a model where the read-only input tape contains the initial input graph G_0, the first read-write work tape contains the input triple $(v_0, (i_0, j_0))$, where v_0 is a vertex of G_i, $i_0 \in [d \cdot c^{i-1}]$ is an edge label in G_i, and $j_0 \in [c]$ is an edge label in H_i, and the other work tapes are blank. When the computation ceases, we want $(v_2, (i_3, j_1)) = \mathrm{Rot}_{G_i}(v_0, (i_0, j_0))$ on the first work tape and the other work tapes to be blank.

Let $\mathrm{Space}(G_i)$ be the amount of space required to compute the rotation map of graph G_i. We will show that for all $i \in [k]$, $\mathrm{Space}(G_i) = \mathrm{Space}(G_{i-1}) + O(\log c)$. Note that $\mathrm{Space}(G_0) = O(\log(nd))$.

Fix $i \in [k]$. We begin with $v_0 \in [n], i_0 \in [d \cdot c^{i-1}]$ and $j_0 \in [c]$ on the first work tape and we want to compute $\mathrm{Rot}_{G_i}(v_0, (i_0, j_0))$. We recursively compute $\mathrm{Rot}_{G_{i-1}}(v_0, i_0)$ so that the work tape now contains (v_1, i_1, j_0). Then we compute $\mathrm{Rot}_{H_i}(i_1, j_0)$ so that the tape now contains (v_1, i_2, j_1). Finally, we compute $\mathrm{Rot}_{G_{i-1}}(v_1, i_2)$ so that the tape now contains (v_2, i_3, j_1).

This requires 2 evaluations of the rotation map of G_{i-1} and one evaluation of the rotation map of H_i. Note that we can reuse the same space for each of these evaluations because they happen in succession. The space needed on top of $\mathrm{Space}(G_{i-1})$ is the space to store edge label j_0, which uses $O(\log c)$ space. So $\mathrm{Space}(G_i) = \mathrm{Space}(G_{i-1}) + O(\log c)$. Since i can be as large as k and $\mathrm{Space}(G_0) = O(\log(n \cdot d))$ we get that for all $i \in [k]$, $\mathrm{Space}(G_i) = O(\log n \cdot d + k \cdot \log c)$. ∎

Corollary VI.5. *Let $M_0, \ldots, M_k$ be the transition matrices of $G_0, \ldots, G_k$ as defined in Lemma VI.4. For all $\ell \in [k]$, given coordinates $i, j \in [n]$, entry i, j of M_ℓ can be computed in space $O(\log n \cdot d + k \cdot \log c)$.*

Proof: Lemma VI.4 shows that we can compute neighbors in the graph G_ℓ in space $O(\log n \cdot d + k \cdot \log c)$. Given coordinates i, j the algorithm initiates a tally t at 0 and computes $\mathrm{Rot}_{G_\ell}(i, q)$ for each q from 1 to $d \cdot c^{\ell-1}$, the degree of G_ℓ. If the vertex outputted by Rot_{G_ℓ} is j, then t is incremented by 1. After the loop finishes, t contains the number of edges from i to j and the algorithm outputs $t/d \cdot c^{\ell-1}$, which is entry i, j of M_ℓ. This used space $O(\log n \cdot d + k \cdot \log c)$ to compute the rotation map of G_ℓ plus space $O(\log(d \cdot c^{\ell-1}))$ to store q and t. So the total space usage is $O(\log n \cdot d + k \cdot \log c) + O(\log(d \cdot c^{\ell-1})) = O(\log n \cdot d + k \cdot \log c)$. ∎

It follows from Corollary VI.5 that when $k = O(\log n)$ for all $i \in [k]$ entries in $I + M_i$ can be computed in space $O(\log n \cdot \log c)$. When we apply it, we will use expanders (from Lemma VI.3) of degree $c = O(\mathrm{polylog}(n))$, so our algorithm computes entries of $I + M_i$ in space $O(\log n \cdot \log \log n)$.

Now we prove that we can multiply matrices in small space.

Lemma VI.6. *Given matrices $M^{(1)}, \ldots, M^{(k)}$ and indices i, j $(M^{(1)} \cdot M^{(2)} \cdot \ldots \cdot M^{(k)})_{ij}$ can be computed using $O(\log n \cdot \log k)$ space where n is the dimension of the input matrices.*

Proof: First we show how to multiply two $n \times n$ matrices in $O(\log n)$ space. Given as input matrices $M^{(1)}, M^{(2)}$

and indices i, j we wish to compute

$$(M^{(1)} \cdot M^{(2)})_{ij} = \sum_{\ell=1}^{n} M_{i\ell}^{(1)} M_{\ell j}^{(2)}$$

We use the fact that n n-bit numbers can be multiplied and added in $O(\log n)$ space (in fact in TC^0 [49], [50]). So for a counter ℓ from 1 to n, the algorithm multiplies $M_{i\ell} \cdot M_{\ell j}$ and adds the result. The counter can be stored with $\log n$ bits and the arithmetic can be carried out in logspace.

To multiply k matrices we recursively split the product into two blocks and multiply each block separately. The depth of the recursion is $\log k$ and each level requires $O(\log n)$ space for a total of $O(\log n \cdot \log k)$ space. ∎

Now we can prove Proposition VI.2.

Proof of Proposition VI.2: Let M_0 be the transition matrix of $G_0 = G$ such that $L = I - M_0$. Clearly a two-way labeling of G can be computed in $O(\log n)$ space by just fixing a canonical way for each vertex to label its incident edges. Set $k = \lceil 6 \log d^2 n^2 \rceil$ and $\mu = 1/30k$. Let $c = \mathrm{polylog}(n)$ be a power of 2 and let $t_i = \log d \cdot c^{i-1}$ for all $i \in [k]$. Note that each t_i is an integer because d and c are powers of 2. Let $H_1, \ldots, H_k$ be c_i-regular graphs on $2^{t_i} = d \cdot c^{i-1}$ vertices, respectively and $\lambda(H_i) \le \mu$ as given by Lemma VI.3. So for all $i \in [k]$, $c_i = \mathrm{poly}(t_i, 1/\mu) = \mathrm{polylog}(n)$. Without loss of generality, we can take $c = \mathrm{polylog}(n) \ge c_i$ for all $i \in [k]$ and make each expander c-regular by adding self loops.

For all $i \in [k]$ let $G_i = G_{i-1} \circledS H_i$ and let M_i be the transition matrix of G_i. By Theorem V.7 we have $I - M_i \approx_\epsilon I - M_{i-1}^2$ for all $i \in [k]$ where $\epsilon = \ln(1/(1-\mu))$. By Corollary V.5 $\lambda(G_k) \le 1/3$ and so by Lemma V.6 we have $I - M_k \approx_{\ln(3/2)} I - J$.

From Theorem IV.1, we get that

$$L^+ \approx_\delta \frac{1}{2}(I - J) + \left(\sum_{i=0}^{k-1} \frac{1}{2^{i+2}} W_i \right) + \frac{1}{2^{k+1}} W_k \quad (3)$$

where for all $i \in [k]$

$$W_i = (I + M_0) \cdot \ldots \cdot (I + M_i)(I - J)(I + M_i) \cdot \ldots \cdot (I + M_0)$$

for $\delta = k \cdot \ln(1/(1-\mu)) + \ln(3/2) < .5$ (using the fact that $e^{-2\mu} \le 1 - \mu$). So we need to compute approximation 3 above in $O(\log n \cdot \log \log n)$ space. There are $k + 2 = O(\log n)$ terms in the sum. Aside from the first term (which is easy to compute) and adjusting the coefficient on the last term, the ith term in the expansion looks like

$$\frac{1}{2^{i+2}}(I + M_0) \ldots \cdot (I + M_i)(I - J)(I + M_i) \cdot \ldots \cdot (I + M_0)$$

which is the product of at most $O(\log n)$ matrices. Corollary VI.5 says that for all $i \in [k]$ we can compute the entries of M_i in space $O(\log n \cdot d + k \log c) = O(\log n \cdot \log \log n)$. Lemma VI.6 says that we can compute the product of $O(\log n)$ matrices in $O(\log n \cdot \log \log n)$ space. By the

composition of space bounded algorithms, each term in the sum can be computed in space $O(\log n \cdot \log \log n) + O(\log n \cdot \log \log n) = O(\log n \cdot \log \log n)$. Then since iterated addition can be carried out in $O(\log n)$ space [49], [50], the terms of the sum can be added using an additional $O(\log n)$ space. Again by the composition of space bounded algorithms, the total space usage is $O(\log n \cdot \log \log n)$ for computing a constant approximation to L^+. ∎

C. A more precise pseudoinverse

Now that we have computed a constant approximation to L^+, we show how to improve the quality of our approximation through iterative methods.

Proposition VI.7. *Let L be the normalized Laplacian of an undirected, regular, aperiodic, multigraph. There is an algorithm such that for every constant $\alpha < 1/2$, given $\tilde{L}_1$ such that $\tilde{L}_1 \approx_\alpha L^+$, computes $\tilde{L}_2$ such that $\tilde{L}_2 \approx_\epsilon L^+$ using space $O(\log n \cdot \log \log n + \log n \cdot \log \log(1/\epsilon))$ where n is the length of the input.*

Proof: By Lemma IV.2, given $\tilde{L}_1 \approx_\alpha L$, we can boost the approximation from α to ϵ by setting $k = O(\log(1/\epsilon))$ and computing

$$\tilde{L}_2 = e^{-\alpha} \cdot \sum_{i=0}^{k} L \cdot (I - e^{-\alpha} \tilde{L}_1 \cdot L)^i$$

By Proposition VI.2, entries in $\tilde{L}_1$ can be computed in space $O(\log n \cdot \log \log n)$ and hence entries in $I - e^{-\alpha} \cdot \tilde{L}_1 \cdot L$ can be computed in space $O(\log n \cdot \log \log n)$ because matrix addition and multiplication can be done in $O(\log n)$ space. Viewing $(I - e^{-\alpha} \cdot \tilde{L}_1 \cdot L)$ as a single matrix, each term in the sum above is the product of at most $i + 1$ matrices. Lemma VI.6 tells us that we can compute such a product in space $O(\log n \cdot \log i)$. Since i can be as large as $k = O(\log 1/\epsilon)$, this gives space $O(\log n \cdot \log \log(1/\epsilon))$ for computing each term. Then since iterated addition can be computed in logarithmic space [49], [50], the $k + 1$ terms in the sum can be added using an additional $O(\log N)$ space where N is the bit length of the computed terms. By composition of space bounded algorithms, computing $\tilde{L}_2$ uses a total of $O(\log n \cdot \log \log n + \log n \cdot \log \log(1/\epsilon))$ space. ∎

VII. PROOF OF MAIN RESULT

Now we are ready to prove our main technical result, Theorem III.1. Theorem I.1 follows as a corollary and is discussed in Section VIII.

Proof of Theorem III.1: Let $L = D - A$ be the input Laplacian of the graph G. Let Δ be the maximum vertex degree in G. We construct a graph G' as follows: at every vertex v, we add $2^{\lceil \log 2\Delta \rceil} - d(v)$ self loops where $d(v)$ denotes the degree of v. Note that $2^{\lceil \log 2\Delta \rceil} - d(v) \ge \Delta$ so this ensures G' is aperiodic and $2^{\lceil \log 2\Delta \rceil}$-regular. Let E be the diagonal matrix of self loops added in this stage.

Then the degree matrix of G' is $D + E = 2^{\lceil \log 2\Delta \rceil} \cdot I$ and the adjacency matrix of G' is $A + E$. So the unnormalized Laplacian of G' is $D + E - A - E = D - A$, which is the same as the unnormalized Laplacian of G. Let $L' = (D - A)(D + E)^{-1} = I - M$ be the normalized Laplacian of G' where M is the transition matrix of G'.

By Proposition VI.2, we can compute a matrix $\tilde{L}_1$ such that $\tilde{L}_1 \approx_\alpha L'^+$ for $\alpha < 1/2$ in space $O(\log n \cdot \log \log n)$. Then we can compute $\tilde{L}_2 \approx_\epsilon L'^+$ by applying Proposition VI.7 using an additional $O(\log n \cdot \log \log n + \log n \cdot \log \log(1/\epsilon))$ space. By composition of space bounded algorithms this yields a $O(\log n \cdot \log \log n + \log n \cdot \log \log(1/\epsilon)) = O(\log n \cdot \log \log(n/\epsilon))$ space algorithm for computing an ϵ-approximation to L'^+. Recalling that $L' = L/2^{\lceil \log 2\Delta \rceil}$ implies that $\tilde{L}_2/2^{\lceil \log 2\Delta \rceil} \approx_\epsilon L^+$ as desired.

∎

VIII. Corollaries

In this section we prove some applications of Theorem III.1.

Definition VIII.1. Let L be the normalized Laplacian of an undirected multigraph and $b \in \text{im}(L)$. Then $\tilde{x}$ is an ϵ-*approximate solution* to the system $Lx = b$ if there is an actual solution x^* such that

$$\|x^* - \tilde{x}\|_L \leq \epsilon \|x^*\|_L$$

where for all $v \in \mathbb{R}^n$, $\|v\|_L \equiv \sqrt{v^\intercal L v}$.

Here we prove that our algorithm for computing an ϵ-approximation to the pseudoinverse of a Laplacian can be used to solve Laplacian systems (Theorem I.1). The following lemma is useful for translating between the L-norm and the ℓ_2 norm.

Lemma VIII.2. *Let $x \in \mathbb{R}^n$ such that $x \perp \vec{1}$ and L be the normalized Laplacian of an undirected multigraph with smallest nonzero eigenvalue γ_2 and largest eigenvalue γ_n. Then*

$$\gamma_2 \|x\|_2^2 \leq \|x\|_L^2 \leq \gamma_n \|x\|_2^2$$

Proof: By definition $\|x\|_L^2 = x^\intercal L x$ and by the variational characterization of the eigenvalues it follows that

$$\gamma_2 \cdot \|x\|_2^2 = \gamma_2 \cdot x^\intercal x \leq \|x\|_L^2 \leq \gamma_n \cdot x^\intercal x = \gamma_n \cdot \|x\|_2^2$$

∎

Since for undirected multigraphs with maximum degree d, $\gamma_2 \geq 1/2dn^2$ and $\gamma_n \leq 2d$ it follows from the above lemma that for all $x \perp \vec{1}$, $\|x\|_2$ and $\|x\|_L$ are within a multiplicative factor of $2d \cdot n$ of one another. We also use the following fact.

Lemma VIII.3. *If $\tilde{L}^+ \approx_\epsilon L^+$, $\epsilon \leq \ln(2)$ and $b \in \text{im}(L)$ then $\tilde{x} = \tilde{L}^+ b$ is a $\sqrt{2\epsilon}$-approximate solution to $Lx = b$.*

A proof of Lemma VIII.3 can be found in the full version of this paper. Now using Theorem III.1 and the lemmas above we can prove Theorem I.1.

Proof of Theorem I.1: Let $L = D - A$ be the input Laplacian of the graph G and let $b \perp \vec{1}$ be the given vector. If G is disconnected then we can use Reingold's algorithm to find the connected components and work on each component separately. So assume without loss of generality that G is connected. We may also assume without loss of generality that G is d-regular. Let $\epsilon' = (\epsilon/(4d^2n^2))^2/2$ where n is the bit length of the input. Theorem III.1 says that we can compute $\tilde{L}^+ \approx_{\epsilon'} L^+$ in space $O(\log n \cdot \log \log(n/\epsilon')) = O(\log n \cdot \log \log(n/\epsilon))$.

By Lemma VIII.3, $x = \tilde{L}^+ b$ is a $\sqrt{2\epsilon'}$-approximate solution to $Lx = b$. In other words, letting $x^* = L^+ b$ we have

$$\|x - x^*\|_L \leq \sqrt{2\epsilon'} \cdot \|x^*\|_L$$

Translating to the ℓ_2 norm we get

$$\frac{1}{2dn} \cdot \|x - x^*\|_2 \leq \|x - x^*\|_L \leq \sqrt{2\epsilon'} \cdot \|x^*\|_L \leq 2dn\sqrt{2\epsilon'} \cdot \|x^*\|_2$$

which implies

$$\|x - x^*\|_2 \leq (2dn)^2 \sqrt{2\epsilon'} \cdot \|x^*\|_2 = \epsilon \cdot \|x^*\|_2$$

as desired.

∎

Our algorithm also implies a $\tilde{O}(\log n)$ space algorithm for computing many interesting quantities associated with undirected graphs. Below we survey a few of these corollaries including hitting times, commute times, and escape probabilities.

Definition VIII.4. In a multigraph $G = (V, E)$ the *hitting time* H_{uv} from u to $v \in V$ is the expected number of steps a random walk starting at u will take before hitting v.

Definition VIII.5. In a multigraph $G = (V, E)$ the *commute time* $C_{uv} = H_{uv} + H_{vu}$ between vertices $u, v \in V$ is the expected number of steps a random walk starting at u will take to reach v and then return to u.

Corollary VIII.6. *Given an undirected multigraph $G = (V, E)$, vertices $u, v \in V$ and $\epsilon > 0$ there are deterministic $O(\log n \cdot \log \log(n/\epsilon))$ space algorithms for computing numbers $\tilde{H}_{uv}$ and $\tilde{C}_{uv}$ such that*

$$|\tilde{H}_{uv} - H_{uv}| \leq \epsilon$$

and

$$|\tilde{C}_{uv} - C_{uv}| \leq \epsilon$$

where n is the bitlength of the input.

Note that since the values we are approximating are at least 1 (if not 0), Corollary VIII.6 also implies that we can achieve relative ϵ error. Our algorithm can also be used to compute escape probabilities.

Definition VIII.7. In a graph $G = (V, E)$, for two vertices u and v, the *escape probability* $p_w(u, v)$ denotes the probability that a random walk starting at vertex w reaches u before first reaching v.

Corollary VIII.8. *Given an undirected multigraph $G = (V, E)$, vertices $u, v \in V$, and $\epsilon > 0$, there is a deterministic $O(\log n \cdot \log \log(n/\epsilon))$ space algorithm for computing a vector $\tilde{p}$ such that*

$$\|\tilde{p} - p\| \le \epsilon$$

where p is the vector of escape probabilities $p_i(u, v)$ in G.

Proofs of Corollaries VIII.6 and VIII.8 can be found in the full version of this paper.

IX. ACKNOWLEDGEMENTS

REFERENCES

[1] R. Impagliazzo and A. Wigderson, "$P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma," in *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, El Paso, Texas, 4–6 May 1997, pp. 220–229.

[2] A. R. Klivans and D. van Melkebeek, "Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses," *SIAM Journal on Computing*, vol. 31, no. 5, pp. 1501–1526 (electronic), 2002. [Online]. Available: http://dx.doi.org/10.1137/S0097539700389652

[3] R. Impagliazzo, V. Kabanets, and A. Wigderson, "In search of an easy witness: exponential time vs. probabilistic polynomial time," *Journal of Computer and System Sciences*, vol. 65, no. 4, pp. 672–694, 2002.

[4] V. Kabanets and R. Impagliazzo, "Derandomizing polynomial identity tests means proving circuit lower bounds," *Computational Complexity*, vol. 13, no. 1-2, pp. 1–46, 2004.

[5] N. Nisan, "Pseudorandom generators for space-bounded computation," *Combinatorica*, vol. 12, no. 4, pp. 449–461, 1992.

[6] M. Saks and S. Zhou, "$\text{BP}_\text{H}\text{SPACE}(S) \subseteq \text{DSPACE}(S^{3/2})$," *Journal of Computer and System Sciences*, vol. 58, no. 2, pp. 376–403, 1999.

[7] R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovász, and C. Rackoff, "Random walks, universal traversal sequences, and the complexity of maze problems," in *20th Annual Symposium on Foundations of Computer Science (San Juan, Puerto Rico, 1979)*. New York: IEEE, 1979, pp. 218–223.

[8] O. Reingold, "Undirected connectivity in log-space," *Journal of the ACM*, vol. 55, no. 4, pp. Art. 17, 24, 2008.

[9] H. Lewis and C. Papadimitriou, "Symmertric space-bounded computation," *Automata, Languages and Programming*, pp. 374–384, 1980.

[10] N. D. Jones, Y. E. Lien, and W. T. Laaser, "New problems complete for nondeterministic log space," *Mathematical Systems Theory*, vol. 10, no. 1, pp. 1–17, 1976.

[11] C. Alvarez and R. Greenlaw, "A compendium of problems complete for symmetric logarithmic space," *Computational Complexity*, vol. 9, no. 2, pp. 123–145, 2000.

[12] O. Reingold, L. Trevisan, and S. Vadhan, "Pseudorandom walks in regular digraphs and the RL vs. L problem," in *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC '06)*, 21–23 May 2006, pp. 457–466, preliminary version as *ECCC* TR05-22, February 2005.

[13] M. Braverman, A. Rao, R. Raz, and A. Yehudayoff, "Pseudorandom generators for regular branching programs," in *FOCS*. IEEE Computer Society, 2010, pp. 40–47.

[14] J. Brody and E. Verbin, "The coin problem and pseudorandomness for branching programs," in *FOCS*. IEEE Computer Society, 2010, pp. 30–39.

[15] M. Koucký, P. Nimbhorkar, and P. Pudlák, "Pseudorandom generators for group products: extended abstract," in *STOC*, L. Fortnow and S. P. Vadhan, Eds. ACM, 2011, pp. 263–272.

[16] J. Síma and S. Zák, "Almost k-wise independent sets establish hitting sets for width-3 1-branching programs," in *CSR*, ser. Lecture Notes in Computer Science, A. S. Kulikov and N. K. Vereshchagin, Eds., vol. 6651. Springer, 2011, pp. 120–133.

[17] P. Gopalan, R. Meka, O. Reingold, L. Trevisan, and S. Vadhan, "Better pseudorandom generators via milder pseudorandom restrictions," in *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS '12)*. IEEE, 20–23 October 2012, to appear.

[18] R. Impagliazzo, R. Meka, and D. Zuckerman, "Pseudorandomness from shrinkage," Electronic Colloquium on Computational Complexity (ECCC), Tech. Rep. TR12-057, 2012.

[19] T. Steinke, "Pseudorandomness for permutation branching programs without the group theory," Electronic Colloquium on Computational Complexity (ECCC), Tech. Rep. TR12-083, July 2012. [Online]. Available: http://eccc.hpi-web.de/report/2012/083/

[20] O. Reingold, T. Steinke, and S. Vadhan, "Pseudorandomness for regular branching programs via Fourier analysis," in *Proceedings of the 17th International Workshop on Randomization and Computation (RANDOM '13)*, ser. Lecture Notes in Computer Science, S. Raskhodnikova and J. Rolim, Eds., vol. 8096. Springer-Verlag, 21–23 August 2013, pp. 655–670, full version posted as ECCC TR13-086 and arXiv:1306.3004 [cs.CC].

[21] T. Steinke, S. Vadhan, and A. Wan, "Pseudorandomness and Fourier growth bounds for width 3 branching programs," in *Proceedings of the 18th International Workshop on Randomization and Computation (RANDOM '14)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), C. Moore and J. Rolim, Eds., 4–6 September 2014, pp. 885–899, full version posted as ECCC TR14-076 and arXiv:1405.7028 [cs.CC], and accepted to *Theory of Computing* Special Issue on APPROX/RANDOM '14.

[22] D. A. Spielman and S.-H. Teng, "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems," in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*. ACM, 2004, pp. 81–90.

[23] R. Peng and D. A. Spielman, "An efficient parallel solver for SDD linear systems," *STOC*, 2014.

[24] E. Rozenman and S. Vadhan, "Derandomized squaring of graphs," in *Proceedings of the 8th International Workshop on Randomization and Computation (RANDOM '05)*, ser. Lecture Notes in Computer Science, no. 3624. Berkeley, CA: Springer, August 2005, pp. 436–447.

[25] M. B. Cohen, J. Kelner, J. Peebles, R. Peng, A. Rao, A. Sidford, and A. Vladu, "Almost-linear-time algorithms for markov chains and new spectral primitives for directed graphs," *arXiv preprint arXiv:1611.00755*, 2016.

[26] M. B. Cohen, J. Kelner, J. Peebles, R. Peng, A. Sidford, and A. Vladu, "Faster algorithms for computing the stationary distribution, simulating random walks, and more," in *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*. IEEE, 2016, pp. 583–592.

[27] K.-M. Chung, O. Reingold, and S. Vadhan, "S-T connectivity on digraphs with a known stationary distribution," *ACM Transactions on Algorithms*, vol. 7, no. 3, pp. Art. 30, 21, 2011, preliminary versions in *CCC '07* and on *ECCC* (TR07-030). [Online]. Available: http://dx.doi.org/10.1145/1978782.1978785

[28] I. Koutis, G. L. Miller, and R. Peng, "Approaching optimality for solving sdd systems. march 2010."

[29] K. Ioannis, G. Miller, and R. Peng, "A nearly-m log n time solver for sdd linear systems," in *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*. IEEE, 2011, pp. 590–598.

[30] J. A. Kelner, L. Orecchia, A. Sidford, and Z. A. Zhu, "A simple, combinatorial algorithm for solving sdd systems in nearly-linear time," in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM, 2013, pp. 911–920.

[31] Y. T. Lee and A. Sidford, "Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems," in *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*. IEEE, 2013, pp. 147–156.

[32] P. Christiano, J. A. Kelner, A. Madry, D. A. Spielman, and S.-H. Teng, "Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs," in *Proceedings of the forty-third annual ACM symposium on Theory of computing*. ACM, 2011, pp. 273–282.

[33] J. A. Kelner, Y. T. Lee, L. Orecchia, and A. Sidford, "An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations," in *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2014, pp. 217–226.

[34] J. A. Kelner and A. Madry, "Faster generation of random spanning trees," in *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*. IEEE, 2009, pp. 13–21.

[35] J. Sherman, "Breaking the multicommodity flow barrier for o (vlog n)-approximations to sparsest cut," in *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*. IEEE, 2009, pp. 363–372.

[36] D. A. Spielman and N. Srivastava, "Graph sparsification by effective resistances," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1913–1926, 2011.

[37] I. Koutis, G. L. Miller, and D. Tolliver, "Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing," in *International Symposium on Visual Computing*. Springer, 2009, pp. 1067–1078.

[38] A. Ta-Shma, "Inverting well conditioned matrices in quantum logspace," in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM, 2013, pp. 881–890.

[39] D. Doron, F. Le Gall, and A. Ta-Shma, "Probabilistic logarithmic-space algorithms for laplacian solvers," *ECCC https://eccc.weizmann.ac.il/report/2017/036/*, 2017.

[40] V. Trifonov, "An o(\logn\log\logn) space algorithm for undirected st-connectivity," *SIAM Journal on Computing*, vol. 38, no. 2, pp. 449–483, 2005.

[41] D. A. Spielman and S.-H. Teng, "Spectral sparsification of graphs," *SIAM Journal on Computing*, vol. 40, no. 4, pp. 981–1025, 2011.

[42] L. Orecchia and N. K. Vishnoi, "Towards an sdp-based approach to spectral methods: A nearly-linear-time algorithm for graph partitioning and decomposition," in *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2011, pp. 532–545.

[43] D. A. Spielman and S.-H. Teng, "A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning," *SIAM Journal on Computing*, vol. 42, no. 1, pp. 1–26, 2013.

[44] Y. T. Lee and A. Sidford, "Efficient inverse maintenance and faster algorithms for linear programming," *CoRR*, vol. abs/1503.01752, 2015. [Online]. Available: http://arxiv.org/abs/1503.01752

[45] O. Reingold, S. Vadhan, and A. Wigderson, "Entropy waves, the zig-zag graph product, and new constant-degree expanders," *Annals of Mathematics*, vol. 155, no. 1, January 2001.

[46] S. P. Vadhan, "Pseudorandomness," *Foundations and Trends® in Theoretical Computer Science*, vol. 7, no. 1–3, pp. 1–336, 2012.

[47] J. Naor and M. Naor, "Small-bias probability spaces: Efficient constructions and applications," *SIAM Journal on Computing*, vol. 22, no. 4, pp. 838–856, Aug. 1993.

[48] N. Alon, O. Goldreich, J. Håstad, and R. Peralta, "Simple constructions of almost k-wise independent random variables," *Random Structures & Algorithms*, vol. 3, no. 3, pp. 289–304, 1992, see also addendum in issue 4(1), 1993, pp. 199–120. [Online]. Available: http://dx.doi.org/10.1002/rsa.3240030308

[49] P. W. Beame, S. A. Cook, and H. J. Hoover, "Log depth circuits for division and related problems," *SIAM Journal on Computing*, vol. 15, no. 4, pp. 994–1003, 1986.

[50] W. Hesse, E. Allender, and D. A. M. Barrington, "Uniform constant-depth threshold circuits for division and iterated multiplication," *Journal of Computer and System Sciences*, vol. 65, no. 4, pp. 695–716, 2002.

[51] *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*. IEEE Computer Society, 2010.