2018 IEEE 59th Annual Symposium on Foundations of Computer Science

Graph Sparsification, Spectral Sketches, and Faster Resistance Computation, via Short Cycle Decompositions

Timothy Chu*, Yu Gao†, Richard Peng†, Sushant Sachdeva‡, Saurabh Sawlani† and Junxing Wang*
*Carnegie Mellon University, Pittsburgh, PA.
Email: tzchu@andrew.cmu.edu, junxingw@andrew.cmu.edu
†Georgia Institute of Technology, Atlanta, GA.
Email: ygao380@gatech.edu, rpeng@cc.gatech.edu, sawlani@gatech.edu
‡University of Toronto, Toronto, ON.
Email: sachdeva@cs.toronto.edu

***Abstract*—We develop a framework for graph sparsification and sketching, based on a new tool, short cycle decomposition – a decomposition of an unweighted graph into an edge-disjoint collection of short cycles, plus a small number of extra edges. A simple observation gives that every graph G on n vertices with m edges can be decomposed in $O(mn)$ time into cycles of length at most $2\log n$, and at most $2n$ extra edges. We give an $m^{1+o(1)}$ time algorithm for constructing a short cycle decomposition, with cycles of length $n^{o(1)}$, and $n^{1+o(1)}$ extra edges. Both the existential and algorithmic variants of this decomposition enable us to make progress on several open problems in randomized graph algorithms.**

1) **We present an algorithm that runs in time $m^{1+o(1)}\varepsilon^{-1.5}$ and returns $(1\pm\varepsilon)$-approximations to effective resistances of all edges, improving over the previous best of $\widetilde{O}(\min\{m\varepsilon^{-2}, n^2\varepsilon^{-1}\})$. This routine in turn gives an algorithm to approximate the determinant of a graph Laplacian up to a factor of $(1\pm\varepsilon)$ in $m^{1+o(1)}+n^{15/8+o(1)}\varepsilon^{-7/4}$ time.**
2) **We show existence and efficient algorithms for constructing graphical spectral sketches – a distribution over graphs H with about $n\varepsilon^{-1}$ edges such that for a fixed vector x, we have $x^\top L_H x = (1\pm\varepsilon)x^\top L_G x$ and $x^\top L_H^+ x = (1\pm\varepsilon)x^\top L_G^+ x$ with high probability, where L is the graph Laplacian and L^+ is its pseudoinverse. This implies the existence of resistance-sparsifiers with about $n\varepsilon^{-1}$ edges that preserve the effective resistances between every pair of vertices up to $(1\pm\varepsilon)$.**
3) **By combining short cycle decompositions with known tools in graph sparsification, we show the existence of nearly-linear sized degree-preserving spectral sparsifiers, as well as significantly sparser approximations of Eulerian directed graphs. The latter is critical to recent breakthroughs on faster algorithms for solving linear systems in directed Laplacians.**

The running time and output qualities of our spectral sketch and degree-preserving (directed) sparsification algorithms are limited by the efficiency of our routines for constructing short cycle decompositions. Improved algorithms for short cycle decompositions will lead to improvements for each of these algorithms.

***Keywords*-graph sketching; graph sparsification, effective resistance; short cycle decomposition;**

Due to space constraints, most of the proofs are deferred to the full version [1] available at http://arxiv.org/abs/1805.12051.

I. Introduction

Graph sparsification is a procedure that, given a graph G, returns another graph H, typically with much fewer edges, that approximately preserves some characteristics of G. Graph sparsification originated from the study of combinatorial graph algorithms related to cuts and flows [2], [3]. Many different notions of graph sparsification have been extensively studied, for instance, spanners [4] approximately preserve pairwise distances, whereas cut-sparsification approximately preserves the sizes of all cuts [2]. Spielman and Teng [5], [6] defined spectral sparsification, a notion that's strictly stronger than a cut-sparsification.

Spectral sparsifiers have found numerous applications to graph algorithms. They are key to fast solvers for Laplacian linear systems [5]–[8]. Recently they have been used as the *sole* graph theoretic primitive in graph algorithms including solving linear systems [9], [10], sampling random spanning trees [11], [12], measuring edge centrality [13], [14], etc.

For an undirected, weighted graph $G = (V, E_G, w_G)$, we recall that the Laplacian of G, L_G is the unique symmetric $V \times V$ matrix such that for all $x \in \mathbb{R}^V$, we have

$$x^\top L_G x = \sum_{(u,v)\in E_G} w_G(u,v)(x_u - x_v)^2.$$

For two positive scalars a, b, we write $a \approx_\varepsilon b$ if $e^{-\varepsilon}a \leq b \leq e^{\varepsilon}a$. We say the graph $H = (V, E_H, w_H)$ is an ε-spectral sparsifier of G if,

$$\forall x \in \mathbb{R}^V, \qquad x^\top L_G x \approx_\varepsilon x^\top L_H x. \tag{1}$$

Restricting the above definition only to vectors $x \in \{\pm 1\}^V$, one obtains cut sparsifiers. For a graph G with n vertices and m edges, Spielman and Teng gave the first algorithm for constructing spectral sparsifiers with $\widetilde{O}(n\varepsilon^{-2})$ edges[1]. Spielman and Srivastava [15] proved that one could construct a sparsifier for G by independently sampling $O(n\varepsilon^{-2}\log n)$ edges with probabilities proportional to their *leverage scores* in G. Finally, Batson, Spielman, and Srivastava [16] proved

[1]The $\widetilde{O}(\cdot)$ notation hides $\text{poly}(\log n)$ factors.

that one could construct sparsifiers with $O(n\varepsilon^{-2})$ edges, and that this is optimal even for constructing sparsifiers for the complete graph. Recently, Carlson *et al.* [17] have proved a more general lower bound, proving that one needs $\Omega(n\varepsilon^{-2}\log n)$ bits to store any data structure that can approximately compute the sizes of all cuts in G.

Given the tight upper and lower bounds, it is natural to guess at this point that our understanding of graph sparsification is essentially complete. However, numerous recent works have surprisingly brought to attention several aspects that we do not seem to understand as yet.

1) Are our bounds tight if we relax the requirement in Equation (1) to hold only for a fixed unknown x with high probability? Andoni *et al.* [18] define such an object to be a *spectral sketch*. They also construct a *data structure* (not a graph) with $\widetilde{O}(n\varepsilon^{-1})$ space that is a spectral sketch for $x \in \{\pm 1\}^V$, even though $\Omega(n\varepsilon^{-2})$ is a lower bound if one must answer correctly for all $x \in \{\pm 1\}^V$. Building on their work, Jambulapati and Sidford [19] showed how to construct such data structures that can answer queries for any x with high probability. A natural question remains open: whether there exist *graphs* that are spectral sketches with $\widetilde{O}(n\varepsilon^{-1})$ edges?
2) What if we only want to preserve the effective resistance[2] between all pairs of vertices? Dinitz, Krauthgamer, and Wagner [20] define such a graph H as a *resistance sparsifier* of G, and show their existence for regular expanders with degree $\Omega(n)$. They conjecture that every graph admits an ε-resistance sparsifier with $\widetilde{O}(n\varepsilon^{-1})$ edges.
3) An ε-spectral sparsifier preserves weighted vertex degrees up to $(1 \pm \varepsilon)$. Do there exist spectral sparsifiers that exactly preserve weighted degrees? Dinitz *et al.* [20] also explicitly pose a related question – does every dense regular expander contain a sparse regular expander?
4) What about sparsification for directed graphs? The above sparsification notions, and algorithms are difficult to generalize to directed graphs. Cohen *et al.* [21] developed a notion of sparsification for Eulerian directed graphs (directed graphs with all vertices having in-degree equal to out-degree), and gave the first almost-linear time algorithms[3] for building such sparsifiers. However, their algorithm is based on expander decomposition, and isn't as versatile as the importance sampling based sparsification of undirected graphs [15]. Is there an easier approach to sparsifying Eulerian directed graphs?
5) There is an ever-growing body of work on the algorithmic applications of graph sparsification [22]–[25]. Could the above improved guarantees lead to even faster algorithms for some of these problems? Two problems of significant interest include estimating determinants [11] and sampling random spanning trees [11], [12], [26].

[2]The effective resistance between a pair u, v is the voltage difference between u, v if we consider the graph as an electrical network with every edge of weight w_e as a resistor of resistance $\frac{1}{w_e}$, and we send one unit of current from u to v.

[3]An algorithm is said to be almost-linear time if it runs in $m^{1+o(1)}$ time on graphs with m edges.

A. Our Contributions

In this paper, we develop a framework for graph sparsification based on a new graph-theoretic tool we call *short cycle decomposition*. Informally, a short cycle decomposition of a graph G is a decomposition into a sparse graph, and several cycles of short length. We use our framework to give affirmative answers to all the challenges in graph sparsification discussed in the previous section. Specifically:

1) We show that every graph G has a graph H with $\widetilde{O}(n\varepsilon^{-1})$ edges that is an ε-spectral-sketch for G. The existence of such graphic spectral-sketches was not known before. Moreover, we give an algorithm to construct an ε-spectral-sketch with $n^{1+o(1)}\varepsilon^{-1}$ edges in $m^{1+o(1)}$ time. In addition, H is also a spectral-sketch for $\boldsymbol{L}_G^{+}$.
2) We show every graph G has an ε-resistance sparsifier with $\widetilde{O}(n\varepsilon^{-1})$ edges, affirmatively answering the question raised by Dinitz *et al.* [20]. We also give an algorithm to construct ε-resistance sparsifiers with $n^{1+o(1)}\varepsilon^{-1}$ edges in $m^{1+o(1)}$ time.
3) We show that every graph has an ε-spectral sparsifier with $\widetilde{O}(n\varepsilon^{-2})$ edges that exactly preserves the weighted-degrees of all vertices. It follows that every dense regular expander contains a sparse (weighted) regular expander. Before our work, it was not known if there exist sparse degree-preserving sparsifiers (even for cut sparsifiers).
4) We show that short cycle decompositions can be used for constructing sparse spectral approximations for Eulerian directed graphs under the notion of spectral approximation given by Cohen *et al.* [21] for Eulerian directed graphs (see Section III-B for definition). We show that short-cycle decompositions are sufficient for sparsifying Eulerian directed graphs, and prove that every directed Eulerian graph has a spectral approximation with $O(n\varepsilon^{-2}\log^4 n)$ edges.
5) We build on our spectral-sketches, to give an algorithm for estimating the effective resistances of all edges up to a factor of $(1 \pm \varepsilon)$ in $m^{1+o(1)}\varepsilon^{-1.5}$ time. The previous best results for this algorithm were $\widetilde{O}(m\varepsilon^{-2})$ [15] and $\widetilde{O}(n^2\varepsilon^{-1})$ [19].

   Incorporating this result into the work of Durfee *et* *al.* [12] gives an $m^{1+o(1)} + n^{15/8+o(1)}\varepsilon^{-7/4}$ time algorithm for approximating the determinant of a $\boldsymbol{L}$ (rather, $\boldsymbol{L}$ after deleting the last row and column, which is the number of spanning trees in a graph), up to a factor of $(1 \pm \varepsilon)$. The previous best algorithm for this problem ran in time $\widetilde{O}(n^2\varepsilon^{-2})$ [11].

As a key component of all our results, we present efficient algorithms for constructing short cycle decompositions. From a bird's eye view, the key advantage provided by short cycle decompositions for all the above results, is that they allow us to sample edges in a coordinated manner so to preserve weighted vertex degrees exactly.

Definition 1. *An $(\widehat{m}, L)$-short cycle decomposition of an unweighted undirected graph G, decomposes G into several edge-disjoint cycles, each of length at most L, and at most $\widehat{m}$ edges not in these cycles.*

The existence of such a decomposition with $\widehat{m}(m,n) \leq 2n$ and $L(m,n) \leq 2\log n$ is a simple observation. We repeatedly remove vertices of degree at most 2 from the graph, along with their incident edges (removing at most $2n$ edges in total). If the remaining graph has no cycle of length at most $2\log n$, a breadth-first search tree of depth $\log n$ starting from any remaining vertex will contain more than n vertices, a contradiction. This can be implemented as an $O(mn)$ time algorithm to find a $(2n, 2\log n)$-short cycle decomposition, which in turn implies a similar running time for all the existential results above. Finding such decompositions faster is a core component of this paper: we give an algorithm that constructs an $(n^{1+o(1)}, n^{o(1)})$-short cycle decomposition of a graph in $m^{1+o(1)}$ time.

Organization: We first define a few necessary preliminaries in Section II. Due to space constraints, we only give an overview of our results in this paper (Section III), and refer the reader to the full version [1] for a detailed exposition. Section III is further divided into the following parts: We start with degree-preserving spectral sparsifiers in Section III-A, and then outline the algorithm for sparsification of Eulerian directed graphs (Section III-B). Next, we sketch the construction of spectral-sketches and resistance sparsifiers in Section III-C, followed by our algorithm for estimating effective resistances for all edges in Section III-D. Finally, we outline our almost-linear time algorithm for constructing a short cycle decomposition in Section III-E.

II. Preliminaries

A square symmetric $n \times n$ matrix $\boldsymbol{M}$ is positive semi-definite (PSD), denoted $\boldsymbol{M} \succeq 0$, if for all $\boldsymbol{x} \in \mathbb{R}^n$, we have $\boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{x} \geq 0$. For two matrices $\boldsymbol{M}_1, \boldsymbol{M}_2$, we write $\boldsymbol{M}_1 \succeq \boldsymbol{M}_2$ if for all $\boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{x}^\top \boldsymbol{M}_1\boldsymbol{x} \geq \boldsymbol{x}^\top \boldsymbol{M}_2\boldsymbol{x}$, or equivalently $\boldsymbol{M}_1 - \boldsymbol{M}_2 \succeq 0$.

For $\varepsilon \geq 0$, and two positive real numbers a, b, we write $a \approx_\varepsilon b$ to express $e^{-\varepsilon}a \leq b \leq e^{\varepsilon}a$. Observe that $a \approx_\varepsilon b$ if $b \approx_\varepsilon a$. For two PSD matrices $\boldsymbol{M}_1, \boldsymbol{M}_2$, we write $\boldsymbol{M}_1 \approx_\varepsilon \boldsymbol{M}_2$ if for all $\boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{x}^\top \boldsymbol{M}_1\boldsymbol{x} \approx_\varepsilon \boldsymbol{x}^\top \boldsymbol{M}_2\boldsymbol{x}$.

Fact 1. *For any PSD $\boldsymbol{M}_1, \boldsymbol{M}_2, \boldsymbol{M}_3$ and $\varepsilon_1, \varepsilon_2 \geq 0$, if we have $\boldsymbol{M}_1 \approx_{\varepsilon_1} \boldsymbol{M}_2$ and $\boldsymbol{M}_2 \approx_{\varepsilon_2} \boldsymbol{M}_3$, then $\boldsymbol{M}_1 \approx_{\varepsilon_1+\varepsilon_2} \boldsymbol{M}_3$.*

For two graphs G_1, G_2, we often abuse notation to write $G_1 \succeq G_2$ to mean $\boldsymbol{L}_{G_1} \succeq \boldsymbol{L}_{G_2}$ and $G_1 \approx_\varepsilon G_2$ to mean $\boldsymbol{L}_{G_1} \approx_\varepsilon \boldsymbol{L}_{G_2}$.

For any PSD matrix $\boldsymbol{M}$, we let $\boldsymbol{M}^+$ denote the Moore-Penrose pseudoinverse of $\boldsymbol{M}$. Thus, if $\boldsymbol{M}$ has an eigenvalues $0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$, with unit-norm eigenvectors $v_1, v_2, \ldots, v_n$ respectively, we have $\boldsymbol{M} = \sum_i \lambda_i v_i v_i^\top$, and $\boldsymbol{M}^+ = \sum_{\lambda_i > 0} \frac{1}{\lambda_i} v_i v_i^\top$. Similarly, we have $\boldsymbol{M}^{1/2} = \sum_i \sqrt{\lambda_i} v_i v_i^\top$, and $\boldsymbol{M}^{+/2} = \sum_{\lambda_i > 0} \frac{1}{\sqrt{\lambda_i}} v_i v_i^\top$.

Our notion of approximation is preserved under inverses:

Fact 2. *For any PSD $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$, and any error $\varepsilon > 0$, we have $\boldsymbol{M}_1 \approx_\varepsilon \boldsymbol{M}_2$ if and only if $\boldsymbol{M}_1^+ \approx_\varepsilon \boldsymbol{M}_2^+$.*

For any u, we let χ_u denote the vector such that the u^{th} coordinate is 1, and all other coordinates are 0. We let $\chi_{uv} = \chi_u - \chi_v$. For any edge $e = (u,v)$ in a connected graph G, the effective resistance of e is defined as $R_{\text{eff}}(e) = \chi_{uv}^\top \boldsymbol{L}_G^+ \chi_{uv}$. For a directed graph $\vec{G}$, its directed Laplacian $\boldsymbol{L}_{\vec{G}}$, can be defined as $\sum_{e=u\to v} \chi_{uv}^\top \chi_u$.

All logarithms throughout the paper are with base 2. Unless mentioned, we assume that our input graph G has m edges and n vertices. Throughout the paper, we consider graphs with positive integral weights on the edges. Whenever we say the weights are poly bounded, we assume they are bounded by $n^{O(1)}$. The expression with high-probability means with probability larger than $1 - \frac{1}{n^{\Omega(1)}}$.

III. Overview

There are 4 major approaches to date towards graph sparsification: expander partitioning [6], [18], [19], importance sampling [2], [15], [27], potential function based [16], [28]–[30], and spanners based, which use sampling via matrix concentration [31]–[33]. A survey of these approaches can be found in [24].

We present a framework for graph sparsification built on short cycle decomposition that merges several ideas from the importance-sampling and spanners based approaches. Before giving an overview of the results in our paper, we first present an alternative algorithm for the classic graph sparsification result of Spielman and Srivastava [15]. This will be quite useful since our algorithms for constructing degree-preserving sparsifiers and sparsifying Eulerian directed graphs are immediately built on the following algorithm, and degree-preserving sparsification is a key idea underlying all our remaining results.

Say we have a graph $G(V,E,w)$ with m edges and n edges. We start by expressing $\boldsymbol{L}_G = \sum_{e\in E} w_e \boldsymbol{L}_e$, where for edge $e = (u, v)$, $\boldsymbol{L}_e = \chi_{uv}\chi_{uv}^\top$. We can re-write this as

$$\boldsymbol{\Pi} = \sum_e w_e \boldsymbol{L}_G^{+/2} \boldsymbol{L}_e \boldsymbol{L}_G^{+/2},$$

where $\boldsymbol{\Pi}$ is the projection orthogonal to the all ones vector. Given a subset of edges $E' \subseteq E$, we draw a random graph H as follows, independently for every edge $e \in E'$, we include it in H with probability $1/2$ and weight $2w_e$. Otherwise, we delete the edge e. All edges $e \in E \setminus E'$ are included in H with weight w_e. Observe that the expectation of $\boldsymbol{L}_H$ is $\boldsymbol{L}_G$.

It follows from standard concentration results for sums of matrix random variables that if for each edge e in E', the norm $\left\| w_e \boldsymbol{L}_G^{+/2} \boldsymbol{L}_e \boldsymbol{L}_G^{+/2} \right\|$ is bounded (up to constants) by $\frac{\varepsilon^2}{\log n}$, then with high probability, $\boldsymbol{L}_H \approx_\varepsilon \boldsymbol{L}_G$.

Now, observe that $\left\| w_{uv} \boldsymbol{L}_G^{+/2} \boldsymbol{L}_{uv} \boldsymbol{L}_G^{+/2} \right\| = w_e \chi_{uv} \boldsymbol{L}_G^{+} \chi_{uv} = w_{uv} R_{\text{eff}}(u, v)$ (this is defined as the leverage score of the edge uv). A simple trace argument implies $\sum_e w_e R_{\text{eff}}(e) = \frac{n-1}{m}$, and hence at least half the edges satisfy $w_e R_{\text{eff}}(e) \le \frac{2n}{m}$. Letting these edges with low leverage score be the set of edges E' we toss random coins for, we obtain that $\boldsymbol{L}_H \approx_{\sqrt{\frac{2n}{m} \log n}} \boldsymbol{L}_G$. Moreover, in expectation, H has at most $\frac{3}{4}m$ edges.

We can repeat the above sparsification roughly $O(\log n)$ times to go down to $O(n\varepsilon^{-2} \log n)$ edges, at each step sparsifying the graph just obtained. By Fact 1, the final approximation error is given by the sum of the error at each sparsification step. Since the number of edges is going down geometrically, the error is increasing geometrically, and hence is dominated by the error at the last step, yielding that the final graph is an $O\left(\sqrt{\frac{2n \log n}{n\varepsilon^{-2} \log n}}\right) = O(\varepsilon)$ spectral-sparsifier for G.

In order to implement this algorithm efficiently, we need to estimate effective resistances for the edges. For the above algorithm, constant factor estimates of the effective resistances suffice (at the cost of changing the constants involved). Spielman and Srivastava [15] showed that one can obtain constant factor estimates for all the edges together in $\widetilde{O}(m)$ time, resulting in a complete running time of $\widetilde{O}(m)$ for the above sparsification algorithm.

A. Degree Preserving Spectral Sparsification

Now, we adapt the above algorithm to leverage a short-cycle decomposition of the graph. Short cycles permit us to sample correlated edges in the graph while still keeping each random sample small in spectral norm. We first use this approach to construct degree-preserving spectral sparsifiers.

We first formally define a degree-preserving sparsifier.

Definition 2 (Degree-Preserving Sparsifier)**.** *A graph $H(V, E')$ is said to be a degree-preserving ε-sparsifier of $G(V, E)$ if*

1) for every $x \in \mathbb{R}^V$, we have, $x^\top \boldsymbol{L}_G x \approx_\varepsilon x^\top \boldsymbol{L}_H x$, and

2) every vertex $u \in V$ has the same weighted degree in G and H, i.e., $\sum_v w_{u,v}^G = \sum_v w_{u,v}^H$.

Given the above algorithm for usual graph sparsification, the main obstacle is that at each sparsification step, the weighted degrees are not preserved. This is where we require our key tool, a short cycle decomposition, which we now formally define.

Definition 3. *For an undirected unweighted graph $G(V, E)$, we say that $\{C_1, C_2, \ldots\}$ is a an $(\widehat{m}, L)$-short cycle decomposition, if $C_1, C_2, \ldots$ are edge-disjoint cycles in G, each C_i is a cycle of length at most L, and $|E \setminus \bigcup_i C_i| \le \widehat{m}$.*

Assuming that we have an efficient algorithm for constructing an $(\widehat{m}, L)$-short cycle decomposition of any given graph, we show the following theorem.

Theorem 1. *Given $\varepsilon \in (0, 1]$, every undirected graph G with poly-bounded weights has a degree-preserving ε-sparsifier with $O(n\varepsilon^{-2} \log^2 n)$ edges. The algorithm* DegreePreservingSparsify*, given our short cycle decomposition algorithm, takes in a graph G and runs in time $m^{1+o(1)}$ and returns a degree-preserving ε-sparsifier of G with $n^{1+o(1)}\varepsilon^{-2}$ edges.*

The following is a brief description of our degree-preserving sparsification algorithm.

Assume first that our graph G is an unweighted graph that has been given to us as the union of disjoint cycles of even length. We sample a random graph H as follows. For each cycle independently, we index the edges in order start from an arbitrary vertex, and perform the following correlated sampling procedure: with probability $1/2$, we keep only the even indexed edges with weight 2, and with probability $1/2$, we keep only the odd indexed edges with weight 2 (see Figure 1). Observe that H has half as many edges as G, and has exactly the same weighted degrees as G. In order to apply matrix concentration, we need to ensure that for each cycle C, the norm $\left\| \boldsymbol{L}_G^{+/2} \boldsymbol{L}_C \boldsymbol{L}_G^{+/2} \right\|$ is at most $\frac{\varepsilon^2}{\log n}$, where $\boldsymbol{L}_C$ is the Laplacian of the cycle C. This norm is easily upper bounded by $\sum_{e \in C} R_{\text{eff}}(e)$.

If instead, G was any arbitrary unweighted graph, we move all the edges with $R_{\text{eff}} \ge \frac{2n}{m}$ to H. Again, by averaging, we still have at least $m/2$ edges remaining. Now, we greedily pick a bi-partition of the vertices of G such that at least half the remaining edges are crossing the cut. We add all the non-crossing edges to H. Now, we utilize an $(\widehat{m}, L)$-short cycle decomposition of G. Thus, all but $\widehat{m}$ edges of G are partitioned into cycles of length at most L. Observe that all the cycle edges crossing the bi-partition, at least must now be in even cycles, each with total R_{eff} bounded by $\frac{2nL}{m}$. Now, independently for each cycle, we pick even or odd edges with probability $1/2$, and add them to H with weight 2. Assuming $m \ge 8\widehat{m}$, H has at most $\frac{15}{16}m$ edges, the same weighted degree as G, and with high

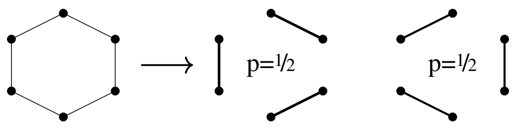

Figure 1: Sampling alternate edges in short cycles for degree-preserving sparsification. The thick edges are double the weights of the thin edges.

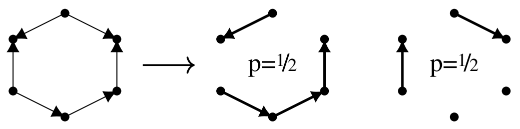

Figure 2: Sampling edges along a random direction in short cycles for sparsification of Eulerian directed graphs. The thick edges are double the weights of the thin edges.

probability $H \approx_{\sqrt{\frac{2nL}{m}\log n}} G$.

Note that re-framing original sparsification into an algorithm for reducing the edges by a constant fraction is crucial for this algorithm. We are only able to reduce the edges in a cycle by half. Further, the cycle decomposition of the graph will necessarily change with every iteration.

For starting with a weighted graph with poly-bounded weights, we can use the binary representation of the edge weight to split each edge into $O(\log n)$ edges, each with a weight that's a power of 2. Now, repeating the above procedure as before, we can construct a degree-preserving ε-sparsifier for G with roughly $\widehat{m}\log n + n\varepsilon^{-2}L\log n$ edges. Using the $O(\log n)$ length short-cycle decomposition, this gives roughly $n\varepsilon^{-2}\log^2 n$ edges.

B. Sparsification of Eulerian Directed Graphs

Now, we can take a very similar approach to sparsifying Eulerian directed graphs. This is a primitive introduced in [21], and is at the core of recent developments in fast solvers for linear systems in directed Laplacians [21], [34], [35]. In contrast to undirected graphs, it has been significantly more challenging to give an appropriate notion of approximation for directed graphs. For a directed graph $\vec{G}$, its directed Laplacian, $\boldsymbol{L}_{\vec{G}}$, can be defined as

$$\boldsymbol{L}_{\vec{G}}(u,v) := \begin{cases} \text{out-degree of } u & \text{if } u = v, \\ -\text{weight}(v \to u) & \begin{array}{l}\text{if } u \neq v \text{ and} \\ v \to u \text{ is an edge.}\end{array} \end{cases}$$

Cohen *et al.* [34] showed that for the purpose of solving linear systems in Eulerian directed graphs, one such useful notion is to say $\vec{H}$ ε-approximates $\vec{G}$ if

$$\left\| \boldsymbol{L}_G^{+/2}(\boldsymbol{L}_{\vec{H}} - \boldsymbol{L}_{\vec{G}})\boldsymbol{L}_G^{+/2} \right\| \leq \varepsilon,$$

where G is the undirectification of G, *i.e.*, the underlying undirected graph of $\vec{G}$ with edge-weights halved. In the case where $\vec{G}$ is Eulerian, $\boldsymbol{L}_G = \frac{1}{2}(\boldsymbol{L}_{\vec{G}} + \boldsymbol{L}_{\vec{G}}^{\top})$.

The key obstacle in sparsifying Eulerian directed graphs is to sample directed subgraphs $\vec{H}$ that are Eulerian since independent sampling cannot provide us with such precise control on the degrees. The work of Cohen *et al.* [34] fixed this locally by modifying the diagonal in $\boldsymbol{L}_{\vec{H}}$ in order to make the sampled graph Eulerian. This approach induces an error in $\boldsymbol{L}_{\vec{H}}$ of the order of $\varepsilon\boldsymbol{D}_{\vec{G}}$ where $\boldsymbol{D}$ is the diagonal out-degree matrix for $\vec{G}$. In order for this error to be small relative to $\boldsymbol{L}_{\vec{G}}$, $\vec{G}$ must be an expander. Hence, the need of expander partitioning in their approach.

However, as we saw above, a short cycle decomposition allows us to perform correlated sampling on edges with precise control on the degrees. For sampling directed graphs, consider a single cycle where the edges may have arbitrary direction (see Figure 2). With probability $1/2$, we sample the edges in clockwise-direction, and with probability $1/2$, we sample the edges in the anti-clockwise direction. In either case, we double the weights of the sampled edges. Observe that for each vertex, the difference between the outgoing and incoming degrees is preserved exactly. Hence, if we started with an Eulerian directed graph, we end up with an Eulerian directed graph. Moreover, in expectation, we only keep half the edges of the cycle.

We can now basically follow the algorithm for degree-preserving sparsification. We treat the graph as undirected for all steps except for sampling edges from a cycle. In particular, the cycle decomposition is found in the corresponding undirected graph. Using the above approach for sampling edges from each cycle, we can sample an Eulerian directed graph, that has a constant fraction fewer edges in expectation. Since the matrices involved are no longer symmetric, we invoke concentration bounds for rectangular matrices to obtain

$$\left\| \boldsymbol{L}_G^{+/2}(\boldsymbol{L}_{\vec{H}} - \boldsymbol{L}_{\vec{G}})\boldsymbol{L}_G^{+/2} \right\| \leq O\left(\sqrt{\frac{nL^3\log n}{m}}\right).$$

Now, repeating this sparsification procedure, and observing that this notion of approximation error also composes, we obtain an Eulerian directed graph $\vec{H}$ that ε-approximates $\vec{G}$ with roughly $\widehat{m}\log n + n\varepsilon^{-2}L^3\log n$ edges. Again, using the naive cycle decomposition, this is $O(n\varepsilon^{-2}\log^4 n)$ edges.

Theorem 2. *Given $\varepsilon \in (0,1]$, for every Eulerian directed graph $\vec{G}$, we can find in $\widetilde{O}(mn)$ time an Eulerian directed graph $\vec{H}$ with $O(n\varepsilon^{-2}\log^4 n)$ edges, that ε-approximates $\vec{G}$.*

This shows the existence of sparsifiers for Eulerian graphs with fewer edges than the nearly-linear sized ones constructed in Cohen *et al.* [34]. More importantly, it shows that approaches based on importance sampling, which work well on undirected graphs, can work in the more general directed settings as well. However, the high costs of computing short cycle decompositions in this paper means this does not lead to faster asymptotic running times in the applications – we believe this is an interesting direction for future work.

C. Graphical Spectral Sketches and Resistance Sparsifiers

We define a graphical spectral sketch as follows:

Definition 4 (Graphical Spectral Sketch)**.** *Given a graph $G(V, E)$, a distribution $\mathcal{H}$ over random graphs $H(V, E')$ is said to be a graphical ε-spectral sketch for G, if for any fixed $\boldsymbol{x} \in \mathbb{R}^V$, with high probability, over the sample $H \sim \mathcal{H}$, we have $\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x} \approx_\varepsilon \boldsymbol{x}^\top \boldsymbol{L}_H \boldsymbol{x}$.*

For constructing graphical spectral sketches, we closely follow the approach of Jambulapati and Sidford [19] and Andoni *et al.* [18]. However, to construct sketches which are graphical, we use an approach similar to the degree-preserving sparsification algorithm. Our result is as follows:

Theorem 3. *Given $\varepsilon \in (0, 1]$, every undirected graph G with n vertices and m edges has a graphical ε-spectral sketch of G with $\widetilde{O}(n\varepsilon^{-1})$ edges. The algorithm* SPECTRALSKETCH*, given G, runs in time $m^{1+o(1)}$, and with high probability returns a graphical ε-spectral sketch of G with $n^{1+o(1)}\varepsilon^{-1}$ edges. In addition, both these graphical sketches satisfy*[4] *that for any fixed $\boldsymbol{x} \in \mathbb{R}^n$, with high probability over the sample $H \sim \mathcal{H}$, we have $\boldsymbol{x}^\top \boldsymbol{L}_G^+ \boldsymbol{x} \approx_\varepsilon \boldsymbol{x}^\top \boldsymbol{L}_H^+ \boldsymbol{x}$.*

First, we define an expander:

The key idea in [19] and [18] is to focus on an expander G, and for each vertex u with degree d_u, sample ε^{-1} edges incident at u and add them to H after scaling its weight by $d_u\varepsilon$ (if $d_u \leq \varepsilon^{-1}$, we add all the edges of u to H), for a total of $n\varepsilon^{-1}$ edges. Firstly, observe that this means that we will have vertices where the degree changes by $\sqrt{\varepsilon^{-1}}$. This is not good enough to preserve $\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x}$ up to $(1 \pm \varepsilon)$ even for the vectors $\boldsymbol{x} \in \{\boldsymbol{\chi}_u\}_{u \in V}$. They get around this by explicitly storing the diagonal degree matrix $\boldsymbol{D}_G$ of G using $O(n)$ extra space. For a fixed vector $\boldsymbol{x}$, they consider the estimator $\boldsymbol{x}^\top \boldsymbol{D}_G \boldsymbol{x} - \boldsymbol{x}^\top \boldsymbol{A}_H \boldsymbol{x}$. Its expectation is easily seen to be $\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x}$. They prove that its standard deviation is bounded by $\varepsilon \cdot O(\boldsymbol{x}^\top \boldsymbol{D}_G \boldsymbol{x})$. For an expander with conductance ϕ, Cheeger's inequality (see [1, Lemma 6.5]) gives that $\varepsilon \cdot \boldsymbol{x}^\top \boldsymbol{D}_G \boldsymbol{x} = \varepsilon \cdot O(\phi^{-2} \boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x})$.

In order to construct an estimator for general graphs, they invoke expander partitioning [6], which guarantees that in any graph G, we can find disjoint vertex induced pieces G, such that each piece is contained in an expander (a well-connected subgraph, formally defined in [1, Section 6.2]), and at least half the edges are contained within such pieces. Applying this $O(\log n)$ times recursively, and combining the above estimators for each piece, Jambulapati and Sidford [19] obtain an estimator with standard deviation $\varepsilon \cdot \widetilde{O}(\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x})$.

[4] $\boldsymbol{L}^+$ denotes the Moore-Penrose pseudo-inverse of $\boldsymbol{L}$.

The above sketch is not a graph since sampling edges does not preserve degrees exactly. Hence, our degree-preserving sparsification algorithm presents a natural approach to convert it into a graphical sketch. We aim to reduce the edge count by a constant factor without incurring too much variance in the quadratic form (and then repeat this process $O(\log n)$ times). We apply expander decomposition, and within each piece, add all edges incident on vertices with degree at most $\widetilde{O}(\varepsilon^{-1}L)$ to H. On the remaining graph, as before, we find a bi-partition, a cycle decomposition, and independently pick odd/even edges in the cycles with double the weight. This reduces the number of edges by a constant factor. Since we preserve weighted degrees *exactly*, an analysis similar to the above gives that for a fixed vector $\boldsymbol{x}$ the standard deviation in $\boldsymbol{x}^\top \boldsymbol{L}_H \boldsymbol{x}$ is bounded by $\varepsilon \cdot \widetilde{O}(\boldsymbol{x}^\top \boldsymbol{L}_G \boldsymbol{x})$. Repeating this process $O(\log n)$ times gives us a graph H with $\widetilde{O}(\widehat{m} + n\varepsilon^{-1}L)$ edges. Averaging $\widetilde{O}(1)$ such sketches, and applying concentration bounds, we obtain a graphical ε-spectral sketch of G.

The fact that we have a graph allows us to reason about the quadratic form of its inverse $\boldsymbol{x}^\top \boldsymbol{L}_H^+ \boldsymbol{x}$. We first argue that H is an $\sqrt{\varepsilon}$-spectral sparsifier of G by showing that the probabilities that we sample edges to form H are good upper bounds of (appropriate rescalings of) effective resistances. This follows because any edge e incident to vertices with degrees at least ε^{-1} that are contained in an expander with expansion at least ϕ has effective resistance at most $O(\phi^{-2}\varepsilon)$.

A simple, but somewhat surprising argument ([1, Lemma 6.8]) gives that if H is a graphical ε-spectral sketch, and a $\sqrt{\varepsilon}$-spectral sparsifier, then for any fixed vector x, with high probability, it also preserves the inverse quadratic form of G, *i.e.*, $\boldsymbol{x}^\top \boldsymbol{L}_H^+ \boldsymbol{x} \approx_{O(\varepsilon)} \boldsymbol{x}^\top \boldsymbol{L}_G^+ \boldsymbol{x}$.

Picking $\boldsymbol{x} \in \{\chi_{uv} | u, v \in V\}$, and taking union bound, we obtain that with high probability, for all $u, v \in V$, $R_{\text{eff}}^G(u, v) \approx_{O(\varepsilon)} R_{\text{eff}}^H(u, v)$. This means that H is a resistance-sparsifier for G with high probability. Again, the naive cycle decomposition gives the existence of resistance-sparsifiers with $\widetilde{O}(n\varepsilon^{-1})$ edges.

Corollary 1 (Resistance Sparsifiers)**.** *For $\varepsilon \in (0, 1]$, every undirected graph G on n vertices has a resistance sparsifier with $\widetilde{O}(n\varepsilon^{-1})$ edges. The algorithm* SPECTRALSKETCH*, given G, runs in time $m^{1+o(1)}$, and with high probability returns a resistance sparsifier of G with $n^{1+o(1)}\varepsilon^{-1}$ edges.*

D. Estimating Effective-Resistances

The effective resistance of an edge is a fundamental quantity. It and its extensions have a variety of connections in the analysis of networks [36], [37], combinatorics [38], [39] and the design of better graph algorithms [26], [40], [41].

While the effective resistance of an edge uv can be computed to high accuracy using linear system solvers, doing so for all edges leads to a quadratic running time. On the other hand, the many algorithmic applications of resistances have motivated studies on efficient algorithms for estimating all resistances. There have been two main approaches for estimating effective resistances to date: random projections [15], [27] or recursive invocations of sparsified Gaussian elimination [11]. Both of them lead to running times of $\widetilde{O}(m\varepsilon^{-2})$ for producing $1 \pm \varepsilon$ estimates of the resistances of all m edges of a graph.

A recent result by Musco *et al.* [42] demonstrated the unlikelihood of high accuracy algorithms (with ε^{-c} dependency for some small c) for estimating the resistances of all edges. On the other hand, the running time of a determinant estimation algorithm for Laplacians by Durfee *et al.* [12] hinges on this ε dependency. The running time bottleneck of this algorithm is the estimation of effective resistances of $\widetilde{O}(n^{1.5})$ edges, but to an multiplicative error of $\varepsilon = n^{-0.25}$. Both methods for estimating resistances described above [11], [15] give running times of $\widetilde{O}(n^2)$ in this setting. Practical studies involving the random projection method for estimating resistances [37], [43] also demonstrate the $\log n\varepsilon^{-2}$ factor in the runtime of such methods translates to solving 10^3 linear systems for a 10% error. Such high overhead has been a major limitation in applying effective resistances to analyzing networks.

A key advantage of our graph sketches and resistance sparsifiers is that because the resulting objects remain as graphs, they can be substituted into the intermediate states of the sparsified Gaussian elimination approach for computing graph sparsifiers [11]. They give a reduction from computing effective resistances to computing approximate Schur complements, which are partial states of Gaussian elimination. Incorporating our spectral sketches in place of generic graph sparsification algorithms with ε^{-2} dependencies gives our main algorithmic result.

Theorem 4. *Given any undirected graph G with n vertices, and m edges, and any t vertex pairs and error $\varepsilon > 0$, we can with high probability compute ε-approximations to the effective resistances between all t of these pairs in $O(m^{1+o(1)} + (n+t)n^{o(1)}\varepsilon^{-1.5})$ time.*

This is the first routine for estimating effective resistances on sparse graphs that obtain an ε dependence better than ε^{-2}. In the dense case an $\widetilde{O}(n^2\varepsilon^{-1})$ result was shown by Jambulapati and Sidford [19], but it relies on $\Omega(n)$ linear systems solves, one per column of the matrix.

We obtain this result via two key reductions:

1) The recursive approximate Gaussian elimination approach from [11] utilizes the fact that effective resistances are preserved under Gaussian eliminations. As this recursion has depth $O(\log n)$, our guarantees for ε-spectral sketches imply that it suffices to work with sketches of such graphs produced by Gaussian elimination. However, Schur complement of very sparse graphs such as the degree n star may have $\Omega(n^2)$ edges. Even if we eliminate an independent set of size $\Theta(n)$, each with roughly average degrees in our spectral sketches with $n\varepsilon^{-1}$ edges, we will end up with at least $n\varepsilon^{-2}$ edges. Thus, we need to directly compute spectral sketches of Schur complements without first constructing the dense graph explicitly.
2) The work of Kyng *et al.* [10] builds fast solvers for Laplacian systems via approximate Cholesky factorization. As a key step, they reduce computing approximating Schur complements to implicitly sparsifying a sum of product weighted cliques [5]. Assuming we start with a spectral-sketch, we know that the graph has total degree $n^{o(1)}\varepsilon^{-1}$, this implies that the total number of vertices involves in these product weighted cliques is $\varepsilon^{-1}n$. Thus, our goal becomes designing an algorithm for implicitly building spectral sketches of product-weighted cliques with a total of $n^{1+o(1)}\varepsilon^{-1}$ vertices that run in time $n^{1+o(1)}\varepsilon^{-(2-c)}$ for some constant $c > 0$.

Our algorithm works with these weighted cliques in time dependent on their representation, which is the total number of vertices, rather than the number of edges. We do so by working with *bi-cliques* as the basic unit, instead of edges. Our algorithm then follows the expander-partitioning based scheme for producing spectral sketches, as in previous works on graph sketches with ε^{-1} type dependencies [18], [19]. This requires showing that this representation as bi-cliques interacts well with both weights and graph partitions. Then on each well-connected piece, we sample $n^{o(1)}\varepsilon^{-0.5}$ matchings from each bi-clique.

This results in each vertex in the bi-clique representation contributing $n^{o(1)}\varepsilon^{-0.5}$ edges to the intermediate sketch. As we are running such routines on the output of spectral sketches, the total number of vertices in these cliques is $n^{o(1)}\varepsilon^{-1}$, giving a total edge count of $n^{1+o(1)}\varepsilon^{-1.5}$. On this graph, we can now explicitly compute another spectral sketch of size $n^{1+o(1)}\varepsilon^{-1}$.

An additional complication is computing an expander decomposition using [1, Lemma 6.7] requires examining all the edges of a graph, which in our case is cost-prohibitive. We resolve this by computing these decompositions on a constant error sparse approximation of this graph instead.

Incorporating this spectral sketch of Schur complements back into [12] gives the first sub-quadratic time algorithm for estimating the determinants of a graph Laplacian with the last row and column removed. This value has a variety of natural interpretations including the number of random spanning trees in the graph. Note that while the determinant may be exponentially large, the result in [12] is stable with variable-precision floating point numbers.

[5]A product weighted clique has a weight vector $\boldsymbol{w}$ with the u, v edge having weight $\boldsymbol{w}_u \boldsymbol{w}_v$.

Corollary 2. *Given any graph Laplacian $\boldsymbol{L}$ on n vertices and m edges, and any error $0 < \varepsilon < 1/2$, we can produce an $1 \pm \varepsilon$ estimate to $\det(\boldsymbol{L}_{-n})$, the determinant of $\boldsymbol{L}$ with the last row/column removed, in time $m^{1+o(1)} + n^{15/8+o(1)}\varepsilon^{-7/4}$.*

Note that the removal of the last row / column is necessary and standard due to $\boldsymbol{L}$ being low rank. Details on this algorithm, and the specific connections with [12] are in [1, Appendix B]. We remark that this algorithm however does not speed up the random spanning tree generation portion of [12] due to it relying on finer variance bounds that require sampling $\widetilde{O}(n^2)$ edges. That spanning tree sampling algorithm however, is superseded by the recent breakthrough result by Schild [26].

E. Almost-Linear Time Short Cycle Decomposition

The bottleneck in the performances of all algorithms outlined above is the computation of short cycle decompositions (Definition 1). The simple existence proof from Section I-A can be implemented to find a short cycle decomposition in $O(mn)$ time (see [1, Section 8] for pseudo-code and proof).

Theorem 5. *The algorithm* NAIVECYCLEDECOMPOSITION*, given an undirected unweighted graph G, returns a $(2n, 2\log n)$-short cycle decomposition of G in $O(mn)$ time.*

While the above algorithm gives us near-optimal length and number of remaining edges[6], we were unable to obtain an almost-linear time algorithm using shortest-path trees. The main obstacle is that updating shortest-path trees is expensive under edge deletions.

Possible Approaches via Spanners: Another approach is to try spanners. The existence of a short cycle decomposition is a direct consequence of spanners. A key result by Awerbuch and Peleg [45] for spanners states that every unweighted graph G has a subgraph H with $O(n)$ edges such that for every edge in $H \setminus G$, its end points are within distance $O(\log n)$ in H. Thus, every edge in $H \setminus G$ is in a cycle of length $O(\log n)$. We can remove this cycle and repeat.

Thus, another approach for generating this decomposition is by dynamic, or even decremental, spanners [46]–[48]. While these data structures allow for poly$(\log n)$ time per update, they are randomized, and crucially, only work against oblivious adversaries. Thus, the update sequence needs to fixed before the data structure samples its randomness. To the best of our understanding, in each of these result, the choice of cycle edges depends upon the randomness. Thus, their guarantees cannot be used for constructing short cycle decompositions. The only deterministic dynamic spanner algorithm we're aware of is the work of Bodwin and Krinninger [49]. However, it has overheads of at least $\sqrt{n}$ in the spanner size / running time.

Possible Approaches via Oblivious Routings: Another possible approach of finding short cycles is via oblivious routings: a routing of an edge e (that doesn't use e) immediately gives a cycle involving e. Since there exist oblivious routings for routing all edges of G in G with small congestion, the average length of a cycle cannot be too large.

Recent works, especially those related to approximate maximum flow, have given several efficient constructions of oblivious routing schemes [50]–[55]. However, such routings only allow us to route a single commodity in nearly-linear time. Using current techniques, routing $\Omega(n)$ arbitrary demands on an expander with poly$(\log n)$ congestion seems to requires $n^{1.5}$ time. On the other hand, on more limited topologies, it is known how to route each demand in sublinear time [56]. Such a requirement of only using local information to route have been studied as myopic routing [57], but we are not aware of such results with provable guarantees.

Our Construction: As an important technical part of this paper, we give an almost-linear-time algorithm for constructing a short cycle decomposition of a graph.

Theorem 6. *The algorithm* SHORTCYCLEDECOMPOSITION*, given an undirected unweighted graph G with n vertices and m edges, returns a $(n^{1+o(1)}, n^{o(1)})$-short cycle decomposition of G in $m^{1+o(1)}$ time.*

Our construction of short cycle decomposition is inspired by oblivious routings, and uses the properties of random walks on expanders. This can be viewed as extending previous works that utilize behaviors of electrical flows [54], [58], but we take advantage of the much more local nature of random walks. This use of random walks to port graphs to fewer vertices is in part motivated by their use in the construction of data structures for dynamically maintaining effective resistances, involving a subset of the authors [59]. It also has similarities with the leader election algorithm for connectivity on well-connected graphs in a recent independent work by Assadi *et al.* [60].

Say we have an expander graph G with conductance ϕ. We know random walks of length $\phi^{-2}\log n$ mix well in G. Choose a parameter k say $n^{1/10}$, and pick the set S of n/k vertices of largest degree (with total degree at least $2m/k$). For every edge e leaving S, starting from its other end point $u \notin S$, we take a $O(\phi^{-2}\log n)$ step random walk. This random walk hits S again with probability $\Omega(1/k)$. Thus, if we pick $k\log n$ random walks, at least one of them will hit S again with high probability. This is a short cycle in $G_{/S}$ (G with S contracted to a single vertex). Since these are independent random walks, Chernoff bounds imply that the maximum congestion is $\widetilde{O}(k\phi^{-2})$. Thus, we can greedily pick a set of $\widetilde{\Omega}(m\phi^4 k^{-1})$ cycles of length $\widetilde{O}(\phi^{-2})$ in $G_{/S}$

[6]Consider the wheel graph with $\frac{n-1}{\log n}$ spokes, and replace each spoke with a path of length $\log n$. This graph has n vertices, $(n-1)\left(1+\frac{1}{\log n}\right)$ edges, and girth of $2\log n + 1$. Lubotzky, Philip, and Sarnak [44] constructed explicit Ramanujan graphs that are 4-regular (and hence have 2n edges) and girth $\frac{4}{3}\log_3 n \geq 0.84\log_2 n$.

that are disjoint.

Now, we just need to connect these cycles within S. We define a graph on the vertices of S, with one edge for every cycle in $G_{/S}$ connecting the two end points in S, and recurse on S. With 10 levels of recursion (since $k = n^{1/10}$), and using the naive cycle-decomposition for the base case, we find a short cycle decomposition in this graph, and then can expand it to a cycle decomposition in G using the cycles in $G_{/S}$. This should give cycles of length $(\phi^{-2} \log n)^{10}$.

There is a key obstacle here: this approach really needs expanders, not pieces contained in expanders, as in the expander decomposition from Spielman and Teng [61]. Instead, we use a recent result of Nanongkai and Saranurak [62] that guarantees the pieces are expanders, at the cost of achieving $\phi = n^{-o(1)}$, and a running time of $m^{1+o(1)}$. A careful trade-off of parameters allows us to recurse for $n^{o(1)}$ iterations, resulting in an $(n^{1+o(1)}, n^{o(1)})$-short cycle decomposition in $m^{1+o(1)}$ time.

IV. SUMMARY

We introduce a framework for graph sparsification based on a short cycle decomposition. Short cycle decomposition allows us to sample edges in a degree preserving manner, which allows us to sparsify graphs while preserving the original degree. This tool enables one to compute a number of sparse graphs efficiently: graph sketches, resistance sparsifiers, degree-preserving sparsifiers, sparsifiers of Eulerian graphs, and more. Critically, this allows one to compute effective resistances on all edges in $O(m^{1+o(1)}\varepsilon^{-1.5})$ time. Our result is the first of its kind to produce a graph that is a resistance sparsifier with $O\left(n^{1+o(1)}/\varepsilon\right)$ edges.

Our algorithms build on the ideas introduced in [18], [19], and a number of other previous works on Eulerian graph sparsification, resistance sparsification, effective resistance computation, determinant estimation, and more. Critically, any improvement to our short-cycle decomposition algorithm will achieve an improvement in all of our results claimed in Section I-A.

Acknowledgment

We thank John Peebles for many insightful discussions, David Durfee for pointing out several key details in the interaction with determinant estimation, and Di Wang for explaining to us the state-of-the-art for efficiently generating expander decompositions.

Yu Gao, Richard Peng and Saurabh Sawlani are supported in part by the National Science Foundation under Grant No. 1718533. Richard Peng is also supported by National Science Foundation under Grant No. 1637523. Sushant Sachdeva's research is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), and a Connaught New Researcher award. La recherche de Sushant Sachdeva est financée par le Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG), et un Connaught bourse pour nouveau chercheurs.

References

[1] T. Chu, Y. Gao, R. Peng, S. Sachdeva, S. Sawlani, and J. Wang, "Graph sparsification, spectral sketches, and faster resistance computation, via short cycle decompositions," *CoRR*, vol. abs/1805.12051, 2018. [Online]. Available: http://arxiv.org/abs/1805.12051 1, 3, 6, 7, 8

[2] A. A. Benczúr and D. R. Karger, "Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, ser. STOC '96. New York, NY, USA: ACM, 1996, pp. 47–55. [Online]. Available: http://doi.acm.org/10.1145/237814.237827 1, 3

[3] D. Eppstein, Z. Galil, G. F. Italiano, and A. Nissenzweig, "Sparsification: a technique for speeding up dynamic graph algorithms," *Journal of the ACM (JACM)*, vol. 44, no. 5, pp. 669–696, 1997. 1

[4] P. Chew, "There is a planar graph almost as good as the complete graph," in *Proceedings of the Second Annual Symposium on Computational Geometry*, ser. SCG '86. New York, NY, USA: ACM, 1986, pp. 169–177. [Online]. Available: http://doi.acm.org/10.1145/10515.10534 1

[5] D. Spielman and S. Teng, "Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems," *SIAM Journal on Matrix Analysis and Applications*, vol. 35, no. 3, pp. 835–885, 2014, available at http://arxiv.org/abs/cs/0607105. 1

[6] D. A. Spielman and S.-H. Teng, "Spectral sparsification of graphs," *SIAM Journal on Computing*, vol. 40, no. 4, pp. 981–1025, 2011, available at http://arxiv.org/abs/0808.4134. [Online]. Available: http://epubs.siam.org/doi/abs/10.1137/08074489X 1, 3, 6

[7] I. Koutis, G. Miller, and R. Peng, "Approaching optimality for solving sdd linear systems," *SIAM Journal on Computing*, vol. 43, no. 1, pp. 337–354, 2014, available at http://arxiv.org/abs/1003.2958. 1

[8] I. Koutis, G. L. Miller, and R. Peng, "A nearly-m log n time solver for SDD linear systems," in *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, ser. FOCS '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 590–598, available at http://arxiv.org/abs/1102.4842. [Online]. Available: http://arxiv.org/abs/1102.4842 1

[9] R. Peng and D. A. Spielman, "An efficient parallel solver for SDD linear systems," in *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, ser. STOC '14. New York, NY, USA: ACM, 2014, pp. 333–342, available at http://arxiv.org/abs/1311.3286. 1

[10] R. Kyng, Y. T. Lee, R. Peng, S. Sachdeva, and D. A. Spielman, "Sparsified cholesky and multigrid solvers for connection laplacians," in *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 2016, pp. 842–850, available at http://arxiv.org/abs/1512.01892. 1, 7

[11] D. Durfee, R. Kyng, J. Peebles, A. B. Rao, and S. Sachdeva, "Sampling random spanning trees faster than matrix multiplication," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 2017, pp. 730–742, available at: https://arxiv.org/abs/1611.07451. 1, 2, 3, 7

[12] D. Durfee, J. Peebles, R. Peng, and A. B. Rao, "Determinant-preserving sparsification of SDDM matrices with applications to counting and sampling spanning trees," in *FOCS*. IEEE Computer Society, 2017, pp. 926–937, available at: https://arxiv.org/abs/1705.00985. 1, 2, 3, 7, 8

[13] H. Li and Z. Zhang, "Kirchhoff index as a measure of edge centrality in weighted networks: Nearly linear time algorithms," in *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2018, pp. 2377–2396, available at: https://arxiv.org/abs/1708.05959. 1

[14] H. Li, R. Peng, L. Shan, Y. Yi, and Z. Zhang, "Current flow group closeness centrality for complex networks," *CoRR*, vol. abs/1802.02556, 2018, available at: https://arxiv.org/abs/1802.02556. 1

[15] D. Spielman and N. Srivastava, "Graph sparsification by effective resistances," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1913–1926, 2011, available at http://arxiv.org/abs/0803.0929. 1, 2, 3, 4, 7

[16] J. Batson, D. A. Spielman, and N. Srivastava, "Twice-Ramanujan sparsifiers," *SIAM Journal on Computing*, vol. 41, no. 6, pp. 1704–1721, 2012. 1, 3

[17] C. Carlson, A. Kolla, N. Srivastava, and L. Trevisan, "Optimal lower bounds for sketching graph cuts," *CoRR*, vol. abs/1712.10261, 2017, available at: https://arxiv.org/abs/1712.10261. 2

[18] A. Andoni, J. Chen, R. Krauthgamer, B. Qin, D. P. Woodruff, and Q. Zhang, "On sketching quadratic forms," in *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*. ACM, 2016, pp. 311–319, available at: https://arxiv.org/abs/1511.06099. 2, 3, 6, 7, 9

[19] A. Jambulapati and A. Sidford, "Efficient $\widetilde{O}(n/\varepsilon)$ spectral sketches for the laplacian and its pseudoinverse," in *SODA*. SIAM, 2018, pp. 2487–2503, available at: https://arxiv.org/abs/1711.00571. 2, 3, 6, 7, 9

[20] M. Dinitz, R. Krauthgamer, and T. Wagner, "Towards Resistance Sparsifiers," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), N. Garg, K. Jansen, A. Rao, and J. D. P. Rolim, Eds., vol. 40. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015, pp. 738–755. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2015/5333 2

[21] 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, available at: https://arxiv.org/abs/1608.03270. 2, 5

[22] D. A. Spielman, "Algorithms, Graph Theory, and Linear Equations in Laplacian Matrices," in *Proceedings of the International Congress of Mathematicians*, 2010, available at: www.cs.yale.edu/homes/spielman/PAPERS/icm10post.pdf. 2

[23] S.-H. Teng, "The Laplacian Paradigm: Emerging Algorithms for Massive Graphs," in *Theory and Applications of Models of Computation*, 2010, pp. 2–14. 2

[24] J. Batson, D. A. Spielman, N. Srivastava, and S.-H. Teng, "Spectral sparsification of graphs: theory and algorithms," *Communications of the ACM*, vol. 56, no. 8, pp. 87–94, Aug. 2013. [Online]. Available: http://doi.acm.org/10.1145/2492007.2492029 2, 3

[25] S. Teng, "Scalable algorithms for data and network analysis," *Foundations and Trends in Theoretical Computer Science*, vol. 12, no. 1-2, pp. 1–274, 2016, available at: http://www-bcf.usc.edu/˜shanghua/teaching/Fall2017-670/networkDataAnalysisPrintedBook.pdf. 2

[26] A. Schild, "An almost-linear time algorithm for uniform random spanning tree generation," *CoRR*, vol. abs/1711.06455, 2017, available at: https://arxiv.org/abs/1711.06455. 2, 6, 8

[27] I. Koutis, A. Levin, and R. Peng, "Improved Spectral Sparsification and Numerical Algorithms for SDD Matrices," in *29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), C. Dürr and T. Wilke, Eds., vol. 14. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012, pp. 266–277, available at http://arxiv.org/abs/1209.5821. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2012/3434 3, 7

[28] Z. Allen Zhu, Z. Liao, and L. Orecchia, "Spectral sparsification and regret minimization beyond matrix multiplicative updates," in *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, 2015, pp. 237–245. 3

[29] Y. T. Lee and H. Sun, "Constructing linear-sized spectral sparsification in almost-linear time," in *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, 2015, pp. 250–269, available at: https://arxiv.org/abs/1508.03261. 3

[30] ——, "An SDP-based algorithm for linear-sized spectral sparsification," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, 2017, pp. 678–687, available at: https://arxiv.org/abs/1702.08415. 3

[31] M. Kapralov and R. Panigrahy, "Spectral sparsification via random spanners," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ser. ITCS '12. New York, NY, USA: ACM, 2012, pp. 393–398, available at http://people.csail.mit.edu/kapralov/papers/spanners.pdf. [Online]. Available: http://doi.acm.org/10.1145/2090236.2090267 3

[32] I. Koutis, “Simple parallel and distributed algorithms for spectral graph sparsification,” in *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures*, ser. SPAA ’14. New York, NY, USA: ACM, 2014, pp. 61–66, available at http://arxiv.org/abs/1402.3851. [Online]. Available: http://doi.acm.org/10.1145/2612669.2612676 3

[33] R. Kyng, J. Pachocki, R. Peng, and S. Sachdeva, “A framework for analyzing resparsification algorithms,” in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2017, pp. 2032–2043, available at: https://arxiv.org/abs/1611.06940. 3

[34] M. B. Cohen, J. Kelner, J. Peebles, R. Peng, A. B. Rao, A. Sidford, and A. Vladu, “Almost-linear-time algorithms for markov chains and new spectral primitives for directed graphs,” in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 2017, pp. 410–419, available at: https://arxiv.org/abs/1611.00755. 5

[35] R. Kyng, “Approximate gaussian elimination,” Ph.D. dissertation, Yale University, 2017, available at: http://rasmuskyng.com/rjkyng-dissertation.pdf. 5

[36] P. Sarkar and A. W. Moore, “A tractable approach to finding closest truncated-commute-time neighbors in large graphs,” in *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2007, pp. 335–343, available at: https://arxiv.org/abs/1206.5259. 6

[37] P. Sarkar, *Tractable algorithms for proximity search on large graphs*. Carnegie Mellon University, 2010, available at: http://www.cs.cmu.edu/˜psarkar/thesis/sarkar_thesis.pdf. 6, 7

[38] L. Lovász, “Random walks on graphs: A survey,” *Combinatorics, Paul Erdős is Eighty*, vol. 2, pp. 1–46, 1993. 6

[39] D. Durfee, M. Fahrbach, Y. Gao, and T. Xiao, “Nearly tight bounds for sandpile transience on the grid,” in *SODA*. SIAM, 2018, pp. 605–624, available at: https://arxiv.org/abs/1704.04830. 6

[40] 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 43rd annual ACM symposium on Theory of computing*, ser. STOC ’11. New York, NY, USA: ACM, 2011, pp. 273–282, available at http://arxiv.org/abs/1010.2921. 6

[41] A. Madry, D. Straszak, and J. Tarnawski, “Fast generation of random spanning trees and the effective resistance metric,” in *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, 2015, pp. 2019–2036, available at http://arxiv.org/pdf/1501.00267v1.pdf. 6

[42] C. Musco, P. Netrapalli, A. Sidford, S. Ubaru, and D. P. Woodruff, “Spectrum approximation beyond fast matrix multiplication: Algorithms and hardness,” in *ITCS*, ser. LIPIcs, vol. 94. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018, pp. 8:1–8:21, available at: https://arxiv.org/abs/1704.04163. 7

[43] C. Mavroforakis, R. Garcia-Lebron, I. Koutis, and E. Terzi, “Spanning edge centrality: Large-scale computation and applications,” in *WWW*. ACM, 2015, pp. 732–742, available at: https://www.cs.cmu.edu/˜jkoutis/papers/spanning_edge.pdf. 7

[44] A. Lubotzky, R. Phillips, and P. Sarnak, “Ramanujan graphs,” *Combinatorica*, vol. 8, no. 3, pp. 261–277, 1988. 8

[45] B. Awerbuch and D. Peleg, “Sparse partitions (extended abstract),” in *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, 1990, pp. 503–513. 8

[46] S. Baswana and S. Sarkar, “Fully dynamic algorithm for graph spanners with poly-logarithmic update time,” in *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, 2008, pp. 1125–1134. 8

[47] A. Bernstein and L. Roditty, “Improved dynamic algorithms for maintaining approximate shortest paths under deletions,” in *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2011, pp. 1355–1365. 8

[48] S. Baswana, S. Khurana, and S. Sarkar, “Fully dynamic randomized algorithms for graph spanners,” *ACM Transactions on Algorithms (TALG)*, vol. 8, no. 4, p. 35, 2012. 8

[49] G. Bodwin and S. Krinninger, “Fully dynamic spanners with worst-case update time,” *CoRR*, vol. abs/1606.07864, 2016, available at: http://arxiv.org/abs/1606.07864. 8

[50] H. Racke, “Optimal hierarchical decompositions for congestion minimization in networks,” in *Proceedings of the 40th annual ACM symposium on Theory of computing*, ser. STOC ’08. New York, NY, USA: ACM, 2008, pp. 255–264. 8

[51] H. Racke, C. Shah, and H. Taubig, “Computing cut-based hierarchical decompositions in almost linear time,” in *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA ‘14, 2014, pp. 227–238. 8

[52] A. Madry, “Fast approximation algorithms for cut-based problems in undirected graphs,” in *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*. IEEE, 2010, pp. 245–254, available at http://arxiv.org/abs/1008.1975. 8

[53] J. Sherman, “Nearly maximum flows in nearly linear time,” in *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, 2013, pp. 263–269, available at http://arxiv.org/abs/1304.2077. 8

[54] 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, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, 2014, pp. 217–226, available at http://arxiv.org/abs/1304.2338. 8

[55] R. Peng, “Approximate undirected maximum flows in $O(m \operatorname{polylog}(n))$ time,” in *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2016, pp. 1862–1867, available at http://arxiv.org/abs/1411.7631. 8

[56] L. G. Valiant, "A scheme for fast parallel communication," *SIAM journal on computing*, vol. 11, no. 2, pp. 350–361, 1982. 8

[57] J. Gao, G. Schoenebeck, and F. Yu, "Cascades and myopic routing in nonhomogeneous kleinberg's small world model," in *WINE*, ser. Lecture Notes in Computer Science, vol. 10660. Springer, 2017, pp. 383–394. 8

[58] J. A. Kelner and P. Maymounkov, "Electric routing and concurrent flow cutting," *Theor. Comput. Sci.*, vol. 412, no. 32, pp. 4123–4135, 2011, available at: https://arxiv.org/abs/0909.2859. 8

[59] D. Durfee, Y. Gao, G. Goranci, and R. Peng, "Fully dynamic effective resistances," *CoRR*, vol. abs/1804.04038, 2018, available at: http://arxiv.org/abs/1804.04038. 8

[60] S. Assadi, X. Sun, and O. Weinstein, "Massively parallel algorithms for finding well-connected components in sparse graphs," *CoRR*, vol. abs/1805.02974, 2018, available at: http://arxiv.org/abs/1805.02974. 8

[61] D. Spielman and S. Teng, "Spectral sparsification of graphs," *SIAM Journal on Computing*, vol. 40, no. 4, pp. 981–1025, 2011, available at http://arxiv.org/abs/0808.4134. [Online]. Available: http://epubs.siam.org/doi/abs/10.1137/08074489X 9

[62] D. Nanongkai and T. Saranurak, "Dynamic spanning forest with worst-case update time: adaptive, las vegas, and $o(n^{1/2-\epsilon})$-time," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, 2017, pp. 1122–1129, available at: https://arxiv.org/abs/1611.03745. 9